UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

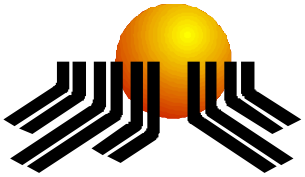
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMO PARA MINERAÇÃO DE OCORRÊNCIAS E FREQÜÊNCIAS DE REGRAS DE ASSOCIAÇÃO TEMPORAIS

EDUARDO JEDLICZKA

ORIENTADOR: PROF.DR. LUIZ CAMOLESI JUNIOR

PIRACICABA, SP
2009



UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMO PARA MINERAÇÃO DE OCORRÊNCIAS E FREQÜÊNCIAS DE REGRAS DE ASSOCIAÇÃO TEMPORAIS

EDUARDO JEDLICZKA

ORIENTADOR: PROF.DR. LUIZ CAMOLESI JUNIOR

Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP
2009

ALGORITMO PARA MINERAÇÃO DE OCORRÊNCIAS E FREQÜÊNCIAS DE REGRAS DE ASSOCIAÇÃO TEMPORAIS

AUTOR: EDUARDO JEDLICZKA

ORIENTADOR: PROF. DR. LUIZ CAMOLESI JUNIOR

Dissertação de Mestrado defendida em 28 de maio de 2009 em Banca Examinadora constituída pelos Professores:

Prof. Dr. Luiz Camolesi Junior – UNIMEP (Orientador)
Prof. Drª. Ana Estela Antunes da Silva- UNIMEP
Prof. Dr. Marco Antonio garcia de Carvalho - UNICAMP

Ao

Autor da vida

À

Minha esposa Julia pela compreensão e motivação

A

Todos que me apoiaram nesta jornada

"Posso não concordar com uma palavra do que dizes,

mas defenderei até o fim o seu direito de dizer."

Voltaire (François-Marie Arouet)

(1694 - 1778)

RESUMO

Muito já foi dito sobre a importância e vantagens da mineração de dados, e a cada dia novas técnicas e ferramentas surgem para obter mais informações e sob novos pontos de vista. O presente trabalho se enquadra nesta última definição, apresentando um algoritmo chamado **AprioriTemporal** que acrescenta a validade temporal ao apriori, resultando num mapa com a freqüência de cada regra encontrada por este.

PALAVRAS-CHAVE: Apriori, Banco de Dados Temporal, Mineração de Dados.

ALGORITHM FOR MINING OCCURRENCE AND FREQUENCY OF TEMPORAL ASSOCIATION RULES

ABSTRACT

Many things have been said about the importance and benefits of data mining, and every day, new tools and techniques emerge to discover more information under a new point of view. This work fits in the last definition, presenting a new algorithm called **AprioriTemporal** where the temporal validity are added to Apriori, resulting in a map with the frequency of each rule (in a given granularity) found by the algorithm.

KEYWORDS: Apriori, Temporal Database, Data Mining

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
BDT	Banco de Dados Temporal
CLT	Consolidação das Leis do Trabalho
CSV	Comma Separated Values
DER	Diagrama Entidade-Relacionamento
DW	Data Warehouse
FK	Foreign Key
GUI	Graphic User Interface
IDE	Integrated Development Environment
KDD	Knowledge Discovery in Database
PIM	Personal Information Manager
SGDB	Sistema Gerenciador de Banco de Dados
SGDB-R	Sistema Gerenciador de Banco de Dados Relacional
SO	Sistema Operacional
UML	Unified Modeling Language

LISTA DE TABELAS

1. INTRODUÇÃO

1.1. CONSIDERAÇÕES INICIAIS

Muito já foi dito sobre a importância e benefícios das várias técnicas de mineração de dados, com o intuito de produzir conhecimento, descobrindo **regras**, **padrões**, **características** e **agrupamentos** visando melhorar a eficiência, ou seja, a maximização dos lucros com redução dos riscos, nos mais variados ramos, entre eles, supermercados, atacadistas, seguradoras, planos de saúde, entidades de crédito e fundos de investimentos.

A mineração de dados é a técnica responsável por gerar conhecimento possibilitando reduzir o impacto do fator sorte na gestão e sucesso destas empresas, pois muitas características (sazonalidades, fins-de-semana ou fim-de-ano, temporadas esportivas, períodos de crescimento econômico, etc..) podem ser descobertas e conhecidas através do uso das ferramentas certas.

Imagine problemas aparentemente óbvios: vender sorvete, ou qualquer outro produto relacionado ao calor, durante o inverno ou em regiões conhecidas por sua baixa temperatura. Sem um estudo detalhado sobre os hábitos de consumo e sobre os potenciais clientes, é possível estar ignorando uma verdadeira mina de ouro.

Veremos que este tipo de informação, descoberto através da mineração de dados, tem um prazo de validade, pois verdades, valores e costumes mudam com o passar das primaveras, e a moda nos ensina que “o belo” de hoje, não o foi ontem, e dificilmente será o mesmo de amanhã, mas pode ter sido o mesmo de nossas avós ou bisavós! Ponderando isto, percebe-se que a tão sonhada maximização da eficiência exige doses cada vez menos “homeopáticas” de “atenção e informações” por parte dos responsáveis pelas

estratégias e administração destas empresas, já que o tempo, o conhecimento das transformações ocorridas com o passar dele, pode ser um ótimo aliado quando utilizado corretamente.

1.2. OBJETIVOS

Dado que o tempo é um assunto pertinente na obtenção de conhecimentos, propomos um novo algoritmo de mineração de regras de associação baseado no **Apriori** (AGRAWAL e SRIKANT 1994) que visa fornecer, além das regras descobertas, sua validade temporal e explicitar um padrão de repetição.

1.3. MOTIVAÇÃO

Atualmente existem bons algoritmos à disposição de usuários e empresas (*FP-Growth*, entre outros), porém conforme explicitado em LI, WANG e JAJODIA, não raro eles apresentam limitações de uso, confiabilidade dos resultados ou baixa eficiência quando aplicados a situações temporais.

Diante da limitação que a variável tempo impõe a alguns destes algoritmos e a pertinência do tema, nos concentramos em sugerir uma nova abordagem visando à descoberta de pontos de vista, pouco abordados por outras técnicas e propostas, confiando que possam fazer parte da próxima geração de ferramentas de mineração disponíveis.

1.4. ORGANIZAÇÃO DO TRABALHO

O presente documento encontra-se dividido em cinco partes: introdução; fundamentação teórica; o algoritmo em si; os resultados obtidos através da experimentação; e a conclusão.

Na fundamentação teórica (capítulo 2), apresentamos alguns conceitos sobre o objeto tempo e sua importância e complexidade, além de uma breve introdução sobre mineração de dados, regras de associação, e o algoritmo Apriori, apresentando também dois trabalhos correlatos.

No terceiro capítulo são demonstrados a estrutura e o funcionamento do algoritmo proposto contemplando também um pequeno exemplo de funcionamento.

O quarto capítulo apresenta: um resumo de como o algoritmo foi implementado; a estrutura do arquivo de entrada; os resultados obtidos através da experimentação em três diferentes bases de dados, com movimentação de dois meses, um ano e quatro anos – todas sintéticas – confrontando seus resultados com o Apriori tradicional.

E por sua vez, o quinto capítulo fecha o presente trabalho, apresentando as contribuições e futuros trabalhos.

2. MINERAÇÃO E REGRAS DE ASSOCIAÇÃO

2.1. CONSIDERAÇÕES INICIAIS

A mineração de dados objetiva a produção de novos conhecimentos garimpando verdades, relacionamentos, regras, semelhanças, no amontoado de dados existentes, sejam em softwares legados, bancos de dados, *PIM*s (gerenciador de contatos/atividades pessoais), arquivos texto ou planilhas eletrônicas, data warehouses, entre outros.

Nigro (Nigro apud Romão *et al* 1999) traduz a filosofia por trás da Mineração de Dados como:

> […] *Mineração de Dados* reúne uma série de técnicas, com destaque para as estatísticas, probabilísticas e de inteligência artificial, capazes de fornecer respostas a várias questões ou mesmo descobrir novas informações em grandes bancos de dados. A *Mineração de Dados* é especialmente útil em casos nos quais não se conhece a pergunta, mas, mesmo assim, existe a necessidade de respostas […]

Laxman e Sastry (2006), numa referência ao controle de informações temporais, explica:

> […] Mineração de Dados preocupa-se com a análise de grandes volumes de dados (muitas vezes não estruturados) visando descobrir regras, padrões ou relacionamentos interessantes que conduzam a uma melhor compreensão dos processos. Diante disto, o campo de atuação de um minerador de Dados Temporal é a obtenção do mesmo tipo de retorno, mas baseando-se, numa massa de dados geralmente ordenada por algum tipo de fator de tempo. […]

Sendo assim, definimos mineração de dados como: conjunto de ferramentas e técnicas aplicadas a uma massa de dados visando produzir

novos conhecimentos, representados através de agrupamentos, associações, regras, etc.

2.2. CONCEITO DE TEMPO

É de domínio público que, assim como todos os equipamentos e máquinas que lhe antecederam, o computador surgiu pela dificuldade dos seres humanos trabalharem com cálculos matemáticos de forma rápida e principalmente livre de erros. No final do século XIX, muito antes de alguém imaginar o nome computador, Hermann Hollerith já utilizava cartões perfurados como uma espécie primitiva de banco de dados realizando o senso americano em tempo recorde.

Nestes quase 120 anos muita coisa mudou, mas estas duas finalidades (guardar informações e realizar cálculos) permanecem inquestionáveis. Partindo deste ponto, não é difícil imaginar que pouco tempo tenha decorrido até que houvesse a real necessidade de, não somente computar, mas controlar e armazenar o tempo ou uma seqüência ordenada de fatos. Laxman e Sastry (2006) nos informam que ainda na primeira metade do século XX já existia a preocupação da influência da variável tempo:

> […] Análise de séries temporais tem uma história relativamente longa. Técnicas para modelagem estatística e análise espectral de séries temporais (reais ou complexas) estão em uso por mais de cinqüenta anos (Box *et al* 1994;.Chatfield 1996). Previsão do tempo, predição e automação de processos (mercado financeiro e acionário) são algumas das mais antigas e estudadas aplicações da análise de séries temporais (Box *et al* 1994) […]

Vários autores já levantaram a temática do controle e representação do tempo (dentre elas a norma ISO 8601), alguns mais focados na melhor forma de armazenagem (objetos complexos, desmembramento, discretização), outros mais focados em sua recuperação (filtros, agrupamentos, paginação), além daqueles que, à exemplo de WHEELWRIGHT e

MAKRIDAKIS 1985; LEVINE et al 2005, demonstram questões estatísticas sobre séries temporais, ou baseadas em períodos ou ciclos temporais como dias, semanas, meses, estações do ano, temporadas esportivas, trimestres, exercícios fiscais, anos letivos, etc.

2.2.1. TEMPO ENQUANTO OBJETO COMPLEXO

Apesar de o tempo ser algo contínuo, sem ajustes ou sobressaltos, e ser completamente transparente e natural para seres humanos, possui alta complexidade quando expresso na forma de calendário ou objeto de cálculo. Na síntese das idéias de CASAS; OLIVIER; e JENSEN *et al* (1994) compreendemos que isto seja reflexo da influência de diferentes eras, culturas e povos, convergindo assim no calendário atualmente em uso nos países cristãos, aprovado em concílio pelo papa Gregório XIII e composto por uma incrível quantidade de regras e medidas. Só para citar alguns:

- Existem muitos calendários, mas **o calendário gregoriano** é dividido em antes e depois de Cristo;

- **a páscoa cristã** acontece entre 22 de março e 25 de abril, ou seja, no primeiro domingo após a lua cheia do equinócio de março (que no hemisfério norte casa com o início da primavera) onde a luz do dia tem igual duração com o período da noite;

- **um ano** pode ser dividido de formas distintas: 12 meses (que se agrupam em **bimestres**, **trimestres**, **quadrimestres** ou **semestres**); 53 ou 54 semanas; 365 ou 366 dias;

- **um ano é bissexto** quando for divisível por 4, exceto se for divisíveis por 100 e não for divisível por 400, por exemplo, 2100 não será bissexto;

- **um mês** pode ter 28, 29, 30 ou 31 dias;

- Uma **semana** tem 7 **dias** de 24 **horas** de 60 **minutos** com 60 **segundos** cada.

- Isto sem citar: feriados, dias úteis ou dias letivos, frações e divisões criadas pelo governo para pagamento de impostos (decendial, quinzenal), trabalhadores noturnos no regime CLT onde, para efeito de cálculo, uma hora tem 52min30s.

Isto posto, concluímos que a tarefa de lidar com o cálculo do tempo e controle das informações temporais não seja das mais simples e agradáveis para analistas e desenvolvedores de produtos de software para computadores.

2.2.2. CLASSIFICAÇÃO DE TEMPO

Existem quatro questões temporais que precisam ser compreendidas sobre o que torna um amontoado de dados e informações (estejam elas numa planilha eletrônica, numa base de dados, num SGDB, ou DW) na sua correlata temporal, apresentados em JENSEN *et al* 1992 e JENSEN *et al* 1994: o tempo instantâneo; o tempo transacional; o tempo de validade; e o tempo bitemporal.

- **Tempo Instantâneo**: é a situação dos dados *in loco*, no momento da consulta, não estando associado a nenhum controle temporal. É o que acontece normalmente quando atualizamos um ou mais registros em um BD, o novo dado sobrepõe o antigo. Por exemplo, se um produto teve vários reajustes de preço nos últimos meses, só sabemos o seu preço atual (consultando diretamente a tabela), e não temos

como consultar qual ou quais eram os seus preços anteriores, quando ocorreram, ou quantas foram estas alterações;

- **Tempo Transacional**: é uma evolução do *tempo instantâneo*, sendo considerado quando um dado, registro ou linha do BD possui uma referência temporal de quando entrou ou saiu do BD (por exemplo: data de cadastro, alteração ou exclusão), seja ela gerada pelo sistema, usuário ou SGDB. E, embora possamos utilizar esta referência temporal (para descobrir quando o produto foi cadastrado ou quando o preço foi alterado), ainda não temos como saber qual era o preço anterior do mesmo;

- **Tempo de Validade** *(também chamado de tempo histórico)*: Diferentemente dos anteriores, ao acontecerem modificações (sejam elas, inclusões, exclusões ou atualizações) serão registradas pelo BDT (qualquer BD com este controle ou estrutura - seja nativa ou através de *triggers*, *packages*, *procedures* - já se caracteriza como temporal) gerando “versões paralelas” destes dados, válidas apenas numa faixa de tempo, possibilitando buscar a situação atual, passada, e qualquer reserva, previsão, ou fato que ocorrerá no futuro (desde que previamente alimentado no BDT). Logo, pode-se buscar os dados, registros e linhas em qualquer momento da linha do tempo. Por exemplo, se for necessário, temos condições de consultar qual era o preço de um produto no dia 28 de janeiro do ano passado, e todas as alterações que ele sofreu/sofrerá (antes e depois disto), ou alimentar o sistema com os novos salários que serão pagos após a homologação do sindicato que só ocorrerá daqui a dois meses sem perder a posição atual do BD. Mas não temos

como saber **quando** foi feita a alteração. Sabemos apenas qual era o valor válido naquela época;

- **Bitemporal**: é a junção do tempo de validade com o tempo transacional. Sendo assim, podemos saber o valor de um determinado dado num momento, qual é a sua validade (quando passou e deixou de ter aquele valor), e quando foi implantado no sistema, e se desejarmos, todas as suas transformações anteriores e posteriores.

2.3. REGRAS DE ASSOCIAÇÃO

Regras de associação são regras do tipo "*A leva a B*" ou "*se A então B*" representadas por **A→B**, onde ambos os lados estão contidos num mesmo conjunto **I** e **A∩B=ø**.

Por exemplo, há algoritmos que consideram esta relação em um mesmo ticket de vendas, enquanto outros buscam relação entre eventos **A** ocorridos num determinado momento e **B** ocorrido tempos depois.

Um **candidato** se torna uma regra válida quando atinge os indicadores mínimos – onde os mais comuns são o suporte e a confiança – que geralmente são fornecidos pelo usuário. Enquanto a aceitação de uma regra está sendo apurada são chamadas de candidato.

O **suporte** é um valor percentual (entre 0% e 100% ou em valores reais entre zero e um) que demonstra a taxa de transações em que aquele candidato específico acontece. Se o suporte calculado for inferior ao informado pelo usuário como mínimo, o canditado é descartado (maiores detalhes serão apresentados ainda neste capítulo). É obtido através da simples divisão da quantidade de ocorrências do candidato pela quantidade total de transações.

De forma semelhante, **confiança** também é um valor percentual (entre 0% e 100% ou em valores reais entre zero e um), porém obtido pela quantidade de ocorrência de todos os itens do candidato (parte **B**) dividido pela quantidade de ocorrência dos itens existentes na parte **A** do candidato.

Entre os algoritmos que adotam regras de associação mais conhecidos podemos citar o Apriori (e suas inúmeras variantes), *depth-first e o FP-Growth*, que serão detalhados a diante.

2.4. ALGORITMOS DE MINERAÇÃO DE REGRAS DE ASSOCIAÇÃO

Seguindo a definição dada por Agrawal *et al* 1993, a mineração por regras de associação consiste num conjunto **I** com n itens (**i)**, um conjunto **D** (database) formado por n transações (**t**) onde cada uma possui um TID (*transaction* idenfier, identificador de transação único) composto por um subconjunto de itens pertencentes a **I**.

2.4.1. APRIORI

Foi proposto em 1994 por AGRAWAL e SRIKANT, e é um dos mais conhecidos e utilizados algoritmos de mineração de dados, e conforme o guia de publicações da IBM era parte projeto QUEST da IBM e seu software *Intelligent Miner*.

Embora existam algoritmos mais antigos para a mesma finalidade, o Apriori se tornou famoso rapidamente pela simplicidade e pelo desempenho excepcional quando comparado com os algoritmos disponíveis na época de seu lançamento (realizando menos leituras no banco de dados, devido à alta eficiência do método de poda na redução dos possíveis candidatos).

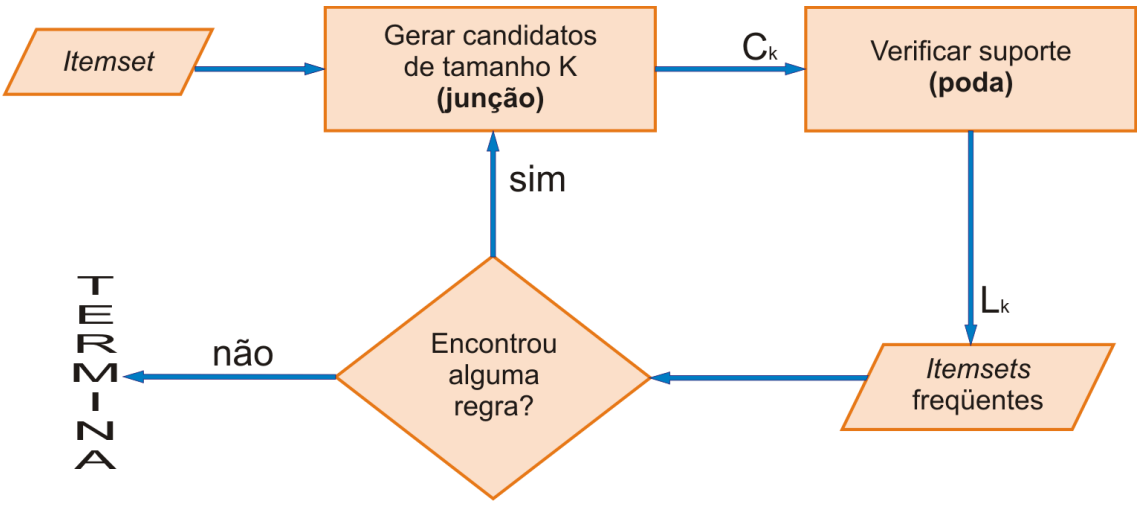


FIGURA 1 - FLUXOGRAMA DO APRIORI

Conforme vemos em Agrawal *et al* 1993, o Apriori utiliza duas variáveis (geralmente fornecidas pelo usuário) chamadas **suporte** e **confiança** para descobrir regras de associação entre os produtos vendidos conjuntamente num mesmo ticket de vendas. Ao analisarmos a figura 1, percebemos que o mesmo consiste em três fases iterativas principais: geração dos candidatos; poda; cálculo do suporte. Sendo iterativas, pois o algoritmo se baseia nas regras descobertas na sua iteração anterior (onde foram descobertas regras de tamanho K-1) para gerar os novos candidatos (de tamanho K), podando-os caso existam em sua composição partes (de tamanho igual ou inferior a K-1) sem o suporte necessário para aceitação da regra. Ele termina quando não foi possível descobrir nenhuma nova regra na iteração anterior.

Na figura 2 vemos um pequeno exemplo do funcionamento do mesmo, onde após a fase de procura inicial em D obtemos os candidatos de tamanho K=1 (C1) e após o cálculo do suporte (responsável pela poda do candidato 4) obtemos (L1). A partir deste ponto o algoritmo se torna recursivo, usando a lista anterior (L1 e L2) para gerar os candidatos (C2 e C3), até que não existam mais regras válidas.

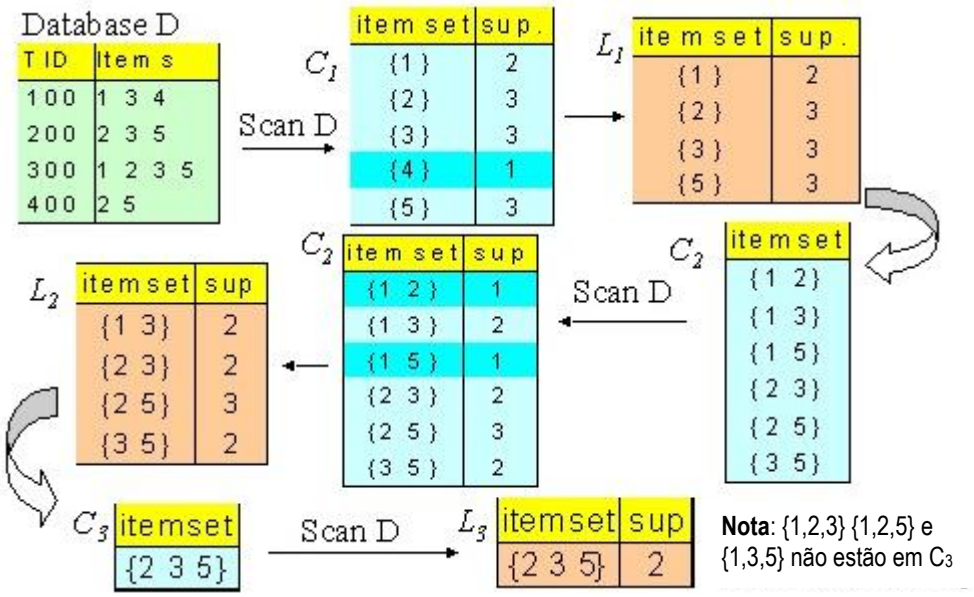


FIGURA 2 - EXEMPLO DE FUNCIONAMENTO DO APRIORI

Após a obtenção de (L_1) foram gerados os candidatos de tamanho K=2 (C_2) realizando uma nova fase de leitura para obter o suporte dos candidatos (C_2) resultando, após a poda, nas regras de tamanho K=2 (L_2), que por sua vez foram utilizadas para gerar os candidatos de tamanho K=3 (C_3), realizando uma nova fase de leitura que, após a poda, resulta nas regras de tamanho K=3 (L_3). Como apenas uma única regra foi encontrada nesta etapa, não será possível gerar candidatos de tamanho K=4, encerrando o algoritmo.

2.4.2. DEPTH-FIRST

SHANG e SATTLER (2005) propuseram um algoritmo chamado Propad (PROjection PAttern Discovery) demonstrando inspiração no *FP-Growth*, cujo grande diferencial em relação a outros algoritmos esta no fato de buscar os dados diretamente num banco de dados relacional via SQL, ao invés de trabalhar com arquivos texto sendo carregados em memória – demonstrando grandes diferenças entre acesso a disco e uso de memória.

Para evitar múltiplas leituras num banco de dados, obtendo melhor desempenho, desenvolveram uma forma de fracionar *itemsets* de tamanho K muito grandes em partes menores – com a deficiência de não poder ser aplicada em tabelas muito esparsas ou muito grandes – através da criação de uma árvore de *itemsets* freqüentes. Além disto, a poda nos candidatos de tamanho K=2 foi drasticamente melhorada, e nos testes apresentados, foram gerados sete candidatos de tamanho K=2 pelo *Depth-First* contra 15 do Apriori.

Num primeiro momento é montada uma tabela denominada *F* contendo os itens e sua quantidade total de ocorrências (obtenção do suporte) das transações existentes na tabela *T (*que possui somente os campos TID e item ordenados de forma crescente), e apenas os itens da transação que atingirem o suporte mínimo são alocados na tabela *T F* (com estrutura semelhante à tabela *T* original).

No passo seguinte são gerados os candidatos de tamanho K=2, onde para cada K=1 (exceto o último elemento) da tabela *T F* é montada uma transação projetada (tabela $PT\__i$) contendo os **candidatos locais** de tamanho K=2 mais freqüentes.

A partir deste ponto, baseado nas tabelas de projeções dos candidatos de tamanho K-1 (para K>2) são geradas, recursivamente, novas tabelas de transação projetada (tabela $PT\__{I\_J}$) apenas com os candidatos locais de tamanho K que atinjam o suporte mínimo até que na conclusão da iteração não sejam encontrados nenhum candidato.

Resumidamente, é gerada uma tabela de transação projetada para cada candidato que atinja o suporte mínimo, assim, a cada iteração, menos dados precisam ser lidos para se concluir aquela etapa.

2.4.3. FP-GROWTH

BORGELT (2005) apresenta uma implementação eficiente do FP-Growth em linguagem C, e compara com outros algoritmos também implementados por ele, como o Apriori, ECLAT e o Relim.

Exceto pelo fato do *FP-Growth* trabalhar com estruturas específicas e especializadas de memória (nas palavras de SHANG e SATTLER 2005) contra tabelas temporárias do *Depth-First*, ele demonstra funcionamento muito semelhante a este, já que ambos se baseiam na eliminação dos candidatos que não atinjam o suporte mínimo para não precisarem repetir a leitura no próximo ciclo.

Utilizando uma estrutura em árvore de padrões freqüentes em ordem decrescente chamada de *FP-Tree, que* é responsável por armazenar a base de dados de forma compacta visando eliminar a quantidade de leituras em disco na fase de geração de candidatos, pois à medida que os candidatos são descartados, também o são das transações que os continham.

2.5. TRABALHOS CORRELATOS

Nos últimos cinco anos, percebemos uma grande quantidade de trabalhos levantando problemas e soluções para questões temporais como problemas de falta de suporte/confiança, detecções de padrões temporais (ciclos utilizando fórmulas de desvio e aproximação ou utilizando múltiplas granularidades), versões especiais de algoritmos tradicionais otimizados para situações temporais. Passamos a detalhar aqui algumas abordagens que influenciaram, por meio de novas idéias e apontando possíveis problemas, diretamente este trabalho.

2.5.1. PADRÕES TEMPORAIS EM MÚLTIPLAS GRANULARIDADES

Sabendo da enorme possibilidade de eventos repetirem-se de tempos em tempos, Li, Wang e Jajodia (2001) propuseram alguns algoritmos: EA – Algoritmo de Enumeração; EAP – Algoritmo de Enumeração com poda entre-níveis; SPA – Algoritmo de ponto de origem; e GBA – Algoritmo baseado em granularidade.

O objetivo era encontrar padrões de repetição temporal que não se enquadrassem numa exata quantidade de tempo, avaliando os candidatos gerados em múltiplas granularidades, pois uma delas poderia tornar mais legível as regras encontradas. Por exemplo, o pagamento de salário não acontece exatamente a cada **x** dias, ou **x** semanas e um padrão de repetição "todo 5º dia útil do mês" é muito mais compreensível do que as equivalentes "acontece entre o 5º e 9º dia de cada mês" ou "se repete entre 26 e 34 dias".

Isto se tornou possível graças a dois fatores, apresentados pelos autores: uma nova terminologia para **padrões simples baseados em calendário** e uma **árvore de calendário**.

Esta nova terminologia foi concebida com definições e regras simples e intuitivas. Como exemplo, podemos citar: **(ano: 2000; mês: *; dia: 1)** - todo dia 1º de qualquer mês do ano 2000; **(semana: *; dia_semana: 1); hora: 11** - às 11h00 de todo domingo; **(mês: *; dia_util_mês: 5)** – todo 5º dia útil do mês.

A estrutura de calendário é uma estrutura em árvore, onde cada nó é uma matriz com um nível de granularidade menor que seu nível superior, possibilitando confrontar ocorrências semelhantes em várias granularidades diferentes até se encontrar um padrão.

2.5.2. MINERAÇÃO POR PADRÕES ESTÁVEIS: INTERVALOS REGULARES ENTRE OCORRÊNCIAS

Graaf e Kosters (2006) apresentaram um algoritmo de mineração denominado *StableCLAT* utilizando uma nova forma de medida de suporte chamada **estabilidade,** cuja função possibilita descobrir uma distância regular (estável) entre eventos e/ou regras que existam em diferentes transações, sendo obtida através da contagem de quantas vezes uma regra/evento acontece entre dois outros eventos em transações diferentes.

Apesar dos autores elogiarem o Apriori e o FP-Growth pela sua eficiência, estes foram descartados enquanto base para modificação devido sua alta complexidade para implementar a *estabilidade* nestes algoritmos. Diante desta dificuldade, os autores tomaram como ponto de partida o ECLAT, que em suas palavras: "…*um algoritmo performático que constrói recursivamente uma matriz de padrões enquanto guarda uma lista de transações que possuem este padrão*…"

O teste de performance e confiabilidade trabalhou principalmente com uma base de dados sintética (baseado em transações de um supermercado), mas também foi utilizado com sucesso em uma base de dados de um web site, demonstrando ser válida em aplicações reais.

2.6. CONSIDERAÇÕES FINAIS

O tema **mineração de dados** é muito amplo, e vai muito além do retratado neste trabalho. Já consolidou–se em governos, órgãos e instituições sem fins lucrativos, além de empresas dos mais variados portes e atividades. Deixou de ser privilégio das grandes corporações, e está sendo incorporado em outras áreas do conhecimento como administração, economia, agricultura, e marketing.

A definição de mineração de dados apresentada por Nigro "… *é especialmente útil em casos nos quais não se conhece a pergunta, mas, mesmo assim, existe a necessidade de respostas*" se demonstra extremamente atual e pertinente, pois sempre imaginamos o que mais poderíamos extrair ou aprender daquela massa de dados acumuladas nos anos de atividade, ou quais vantagens competitivas podem estar inertes, quais perspectivas ou provas de conceito ainda não foram descobertos, quais ciclos de epidemias ou doenças poderão ser antevistos.

Temos a opinião que a validade e classificação temporal e seus desdobramentos já apresentam avanços interessantes, como supracitados nos trabalhos relacionados, mas ainda existe uma enorme lacuna para pesquisa e inovação se comparado aos outros assuntos abordados pela mineração de dados.

3. ALGORITMO PROPOSTO

3.1. CONSIDERAÇÕES INICIAIS

Ao analisar a forma de funcionamento do Apriori, percebe-se que podemos obter resultados equivocados em alguns casos. Numa situação hipotética imaginemos um grande supermercado com muitos anos de funcionamento e registro de vendas. Com o passar do tempo provavelmente ocorreram alterações nos fornecedores e produtos comercializados. Novas marcas surgiram, alguns produtos saíram de linha e talvez até o próprio público alvo tenha se alterado ou ainda mudado de hábitos.

Se utilizarmos esta base em sua totalidade, possivelmente os produtos recentemente lançados que estão sendo um sucesso comercial não conseguirão o suporte mínimo necessário para sua detecção, e ao mesmo tempo, produtos que já saíram de linha continuarão em evidência.

Além disto, está se tornando cada vez mais comuns que as redes de supermercado concentrem as promoções de um tipo de produto num dia específico da semana, como “terça-feira verde” (promoções em frutas, legumes e hortaliças em geral), “quarta-feira vermelha” (promoções em carnes e peixes), “sexta-feira branca” (promoções em pães, bolos, e massas), e chocolates, sobremesas e supérfluos aos sábados e/ou domingos, o que pode gerar ligeiras distorções nos resultados quando comparados na íntegra ou somente naquele dia da semana.

De certo modo, podemos contornar estas duas limitações (bases muito extensas e/ou muito antigas; produtos que são vendáveis especificamente num dia da semana) gerando versões parciais dos dados e repetindo assim inúmeras vezes o processo de mineração até encontrar algo novo.

Além destas duas limitações, ocorre também uma variação da primeira, que reflete a sazonalidade de certos produtos e hábitos dos consumidores, que apesar de também serem detectadas com o Apriori (e outras variantes) mediante a já citada técnica de fracionar o arquivo em diferentes partes, acreditamos que isto descaracterize completamente o objetivo da mineração (descobrir respostas sem conhecer muito bem a pergunta conforme Nigro apud Romão *et al* 1999), pois a tarefa fica extremamente dependente da intuição de quem propõe a divisão.

Sendo assim, com o intuito de superar estas limitações de forma simples, lógica e intuitiva, e acreditando que as mesmas futuramente poderão ser facilmente transplantadas para outros algoritmos, propusemos algumas extensões ao Apriori. Como alteração mais importante, a adoção de uma data de referência em cada transação (aspecto temporal), visando descobrir a validade e repetição de cada regra descoberta em diferentes granularidades/períodos. Cumprindo assim nosso objetivo de detectar não somente a ocorrência da regras, mas também a freqüência que estas acontecem numa janela temporal.

3.2. O ALGORITMO

O algoritmo foi concebido visando acrescentar relevância temporal ao Apriori, buscando descobrir em quais momentos as regras encontradas são válidas, sem se preocupar com o desempenho do mesmo, porém focando-se na descoberta de um novo ponto de vista. A partir desta premissa foi escolhido o Apriori como base pela sua baixa complexidade e ser amplamente conhecido por parte de estudantes, empresas e pesquisadores.

Após uma série de testes e experimentos, chegou-se a conclusão que tratar regras e granularidade em separado seria mais proveitoso, ao simplificar a compreensão e funcionamento do mesmo, resultando assim em três módulos muito claros e bem definidos:

- **definição da granularidade:** controla qual é a faixa temporal adotada em cada iteração e quantas iterações serão necessárias para concluir o trabalho;

- **descoberta de regras** a partir dos *itemsets* pertencentes a cada iteração;

- **consolidação** do mapa de regras e apresentação do resultado.

Com esta modularização ganha-se em flexibilidade, pois alterações em algum destes três módulos pode dispensar ou requerer apenas poucas alterações nos demais.

Assim, torna-se possível acrescentar novos recursos na primeira parte sem precisar alterar o resto do algoritmo. Por exemplo, ao incluir uma nova granularidade ou um tratamento especial para feriados e datas comemorativas.

Também possibilita facilmente a troca do Apriori, que se encontra praticamente intacto no presente trabalho, por qualquer outro. Seja pela necessidade de um melhor desempenho, seja para acrescentar características específicas não necessariamente relacionadas com mineração por regras de associação.

Além de ser possível alterar a forma de consolidação e saída do algoritmo para que o mesmo seja utilizado como matéria-prima para outros algoritmos e técnicas, principalmente na geração de tabelas-verdade, séries estatísticas (previsões e regressões), etc.

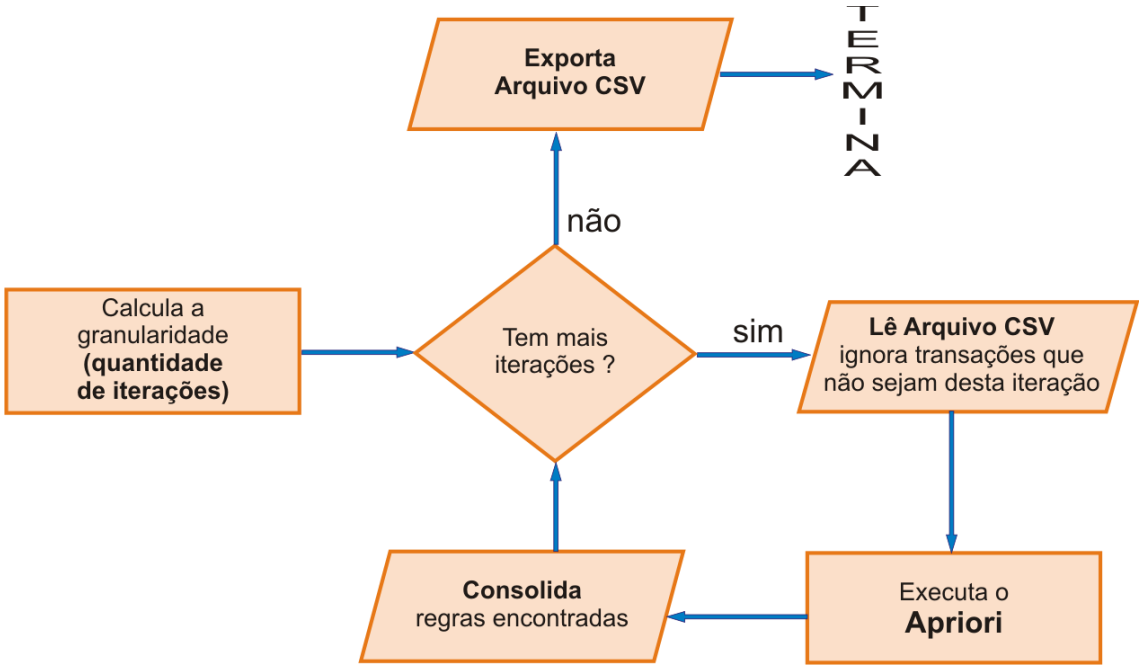


FIGURA 3 – FLUXOGRAMA DO ALGORITMO PROPOSTO

Apresentamos assim, as figuras 3, 4 e 5 contendo respectivamente o fluxograma, sua codificação em português estruturado e um exemplo de funcionamento, definido como semanal com ciclo falso.

```
1)  Calcule a quantidade de iterações;
2)  Crie um mapa de regras vazio;
3)  Para cada iteração faça:
4)      Abra arquivo de transações;
5)      Crie um conjunto de transações
6)      Para cada transação no arquivo faça
7)          Se a data da transação pertence a esta iteração, adicione ao conjunto
        de transações;
8)      Feche o arquivo de transações
9)      Execute o Apriori com o conjunto de transações obtido;
10)     Para cada regra encontrada faça:
11)         Se a regra não existir no mapa, incluir com quantidades zeradas;
12)         Sinalizar a ocorrência da regra no mapa para esta iteração;
13) Exportar os resultados
```

Figura 4 – Algoritmo em português estruturado

Na figura 5, destacamos do lado esquerdo (dentro do quadro cinza claro) a representação de todos os *itemsets* do arquivo CSV (*Comma Separated Values* – porém no nosso caso usando o ponto-e-vírgula como separador), onde o arquivos é totalmente lido durante cada iteração filtrando apenas o que é pertinente para a iteração atual. Neste exemplo, agrupados pelo dia da semana.

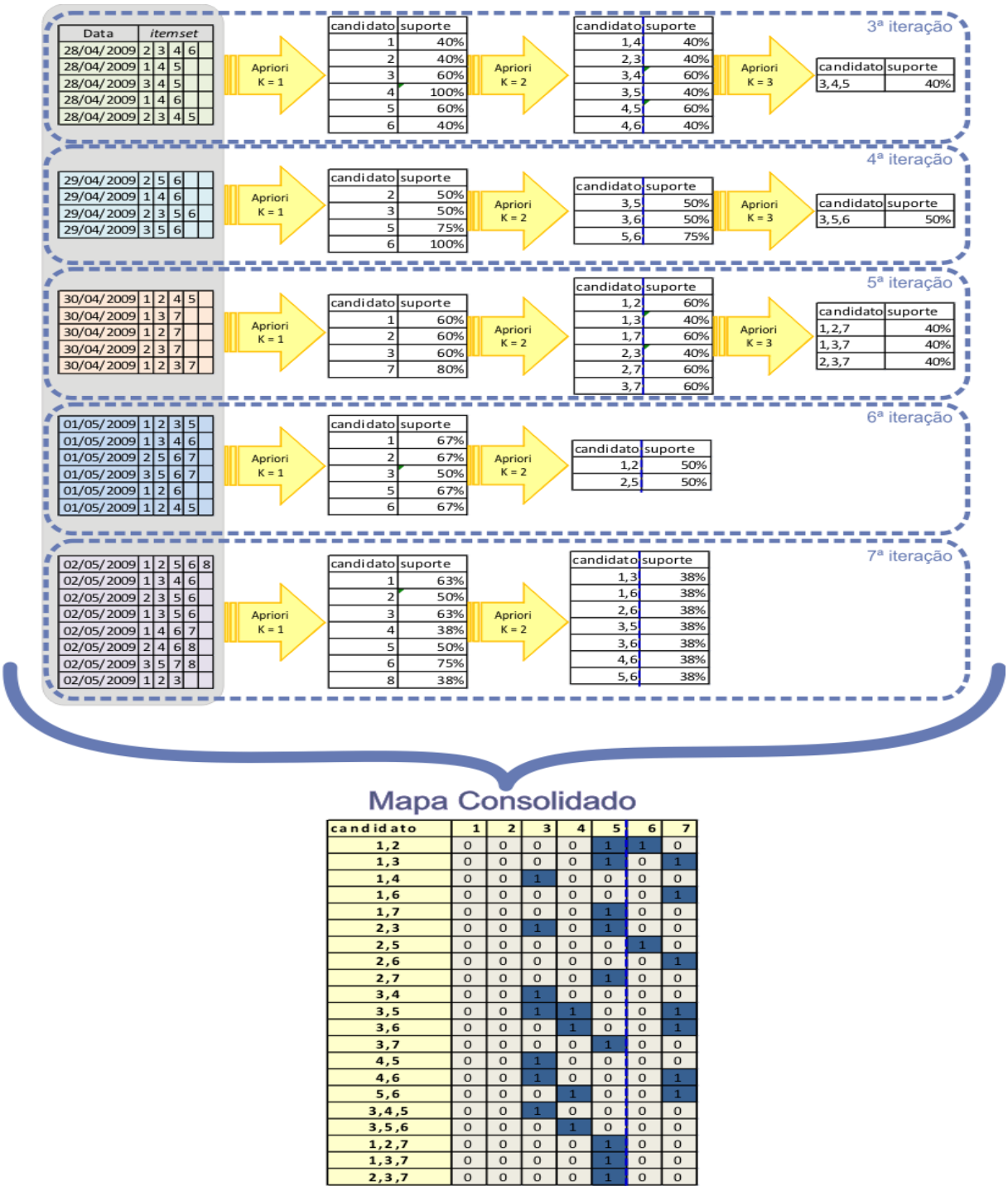


Figura 5 – Exemplo de funcionamento do Algoritmo

Após a execução do Apriori em cada uma destas, é realizada a consolidação das regras e a exportação do mapa, com o indicativo das regras e sua validade.

Obtendo desta forma o mapa, contendo as regras de associação encontradas, exibidas pela coluna candidato, e em quais iterações elas se mostraram válidas. Por estarem agrupadas pelo dia da semana, num arquivo com apenas 5 dias, não há regras válidas para as duas primeiras iterações, enquanto que nenhuma regra se mostrou válida para todos os dias.

3.3. FORMATO DOS RESULTADOS

Existem duas diferentes saídas para o algoritmo. A primeira é válida apenas quando o ciclo é definido como falso, e a granularidade é definida como ZERO – o que força seu funcionamento como o Apriori Tradicional – apresentando na saída apenas os candidatos com seu respectivo percentual de suporte.

TABELA 1- EXEMPLO DE SAÍDA DO APRIORI POR SEMANA DO ANO.

Validade de cada regra por semana										
Candidato	**01**	**02**	**03**	**...**	**49**	**50**	**51**	**52**	**53**	**54**
11, 24	0	0	0	...	0	1	0	0	0	0
12, 17	0	0	0	...	0	0	0	0	1	0
14, 17	0	0	0	...	0	0	0	0	1	0
14, 24	0	0	0	...	0	0	0	1	0	0
14, 28	0	0	0	...	0	0	0	0	1	0
14, 29	0	0	0	...	0	0	0	0	1	0
15, 19	0	0	0	...	0	0	0	1	0	0
15, 24	0	0	0	...	0	0	1	1	0	0
15, 26	0	0	0	...	0	0	0	1	0	0
18, 28	0	0	0	...	0	0	0	1	0	0
19, 24	0	0	0	...	0	0	0	1	0	0
21, 24	0	0	0	...	0	1	0	0	0	0
4, 24	0	0	0	...	0	1	0	0	0	0
Total de regras	**0**	**0**	**0**	...	**0**	**3**	**1**	**6**	**4**	**0**

Nos demais casos retornam-se um arquivo CSV contendo na primeira coluna o candidato entre aspas e *n*-colunas contendo o valor **um** ou **zero**, representando a validade ou não daquela regra naquela janela temporal. Baseado neste arquivo CSV, foi obtida a tabela 1 após a formatação dos campos e o acréscimo do título (neste caso o número da semana) e do rodapé contendo o total de regras descoberto por dia da semana (obtido pelo comando *soma*). Propositadamente, foram omitidas 45 semanas (da 4ª semana até a 48ª semana) por não apresentarem nenhuma regra no período.

3.4. CONSIDERAÇÕES FINAIS

O presente algoritmo fornece uma nova perspectiva ao encontrar regras e explicitar sua validade temporal, tornando-o assim utilizável em diferentes circunstâncias. Acreditamos que estas informações poderão ser utilizadas por gerentes e administradores para entender as mudanças nos padrões de consumo de seus clientes, com o intuito de fidelizar o cliente oferecendo o que ele procura sem perder muito tempo vagueando pelo estabelecimento, ou oferecer promoções nas melhores datas, visando alavancar indiretamente as vendas de outros produtos, ou “desovar” produtos que estejam próximos de seu prazo de validade.

Não obstante, uma análise cuidadosa dos mapas gerados em diferentes granularidades, permitirá encontrar certos padrões e ciclos nas validades das regras, servindo de ótima base para decisões gerenciais ou análise de risco.

E, conforme veremos no próximo capítulo há uma considerável diferença na quantidade e qualidade de regras que podem ser obtidas com o algoritmo proposto se comparado com outros que ignoram a questão temporal.

4. IMPLEMENTAÇÃO E EXPERIMENTAÇÃO

4.1. CONSIDERAÇÕES INICIAIS

Para simplificar o processo de exportação e a visualização, foi adotado o arquivo CSV que pode ser facilmente visualizado e editado em várias planilhas eletrônicas – como o Microsoft Excel e o OpenOffice Calc – que comumente também possibilitam importar dados de diversas fontes, bases e bancos de dados (através de arquivos de largura-fixa, acesso nativo a *DBF*s, acesso via *ODBC*, etc.)

4.2. IMPLEMENTAÇÃO

Conforme supracitado, tomando como ponto de partida o algoritmo Apriori, foi realizada sua completa implementação através da classe TApriori, recebendo como entrada um mapa de todas as transações e seus respectivos conjuntos de itens, e retornando um mapa contendo as regras e seus percentuais de suporte, confiança além da sua participação numérica. Em outras palavras, diferentemente do que é comumente encontrado no mercado nas várias implementações do Apriori tradicional baseadas num arquivo de entrada com saída em tela ou arquivo, a versão apresentada realiza todas as suas operações em memória – apesar da clara vantagem sobre a redução de acesso ao disco e por conseqüência melhor desempenho, constatou-se um alto consumo de memória. Em alguns casos foram gastos mais de 150 MB para uma base sintética com menos de 60 mil transações, estimamos que em condições reais de uso alcance facilmente valores acima de 1 GB de memória.

Como esta classe não inclui informações temporais, podemos utilizá-la como substituto ao Apriori tradicional, o que facilita muito a comparação e compreensão da nova proposta.

Após a concepção desta classe, nos preocupamos com o aspecto temporal, percebendo que diferentes granularidades poderiam levar a diferentes resultados. A necessidade de se agrupar dados em dias, semanas, meses e anos, exigiu a construção de uma classe para padronizar diferentes operações e cálculos envolvendo o tempo, o que deu origem à classe TTempo, responsável por armazenar e converter diferentes medidas e formas de tempo entre si, contendo métodos para importar e exportar strings nos formados DD/MM/YYYY e YYYY-MM-DD, além de possibilitar calcular algumas informações como a diferença entre dias, semanas, meses ou ano entre duas datas.

Para interligar estas duas classes, foi implementada a classe TAprioriTemporal, cuja função é classificar os *itemsets* conforme sua informação temporal (granularidade) e executar a classe TApriori, obtendo assim as regras e sua importância/validade temporal.

Esta operação funciona como um filtro temporal, já que em cada iteração é: (i) realizada a completa leitura do arquivo em disco (visando balancear o consumo de memória); (ii) gerado um mapa de *itemsets* contendo apenas os dados referentes àquele ciclo e granularidade; (iii) executado o TApriori com este mapa; (iv) processado um *merge* das regras descobertas para construir a tabela de existência e validade das diferentes regras em cada ciclo ou fração da granularidade.

As propriedades ciclo e granularidade definem o comportamento e quantidade de iterações geradas. Quando a propriedade ciclo é falsa, é produzida uma iteração para cada membro da granularidade desejada (para granularidade mensal serão 31 iterações já que um mês tem até 31 dias; se for semanal serão 7 (sete) iterações, pois uma semana é composta de 7 dias; como um ano pode ser fracionado em até 54 semanas ou 12 meses, ambas as granularidades foram disponibilizadas. Caso o ciclo seja verdadeiro, as informações serão fracionadas em n períodos de tempo com a duração da granularidade fornecida.

Por exemplo, num supermercado que faça diferentes promoções durante a semana (frutas e legumes na terça-feira, carnes na quarta e pães e massas na sexta), poderia utilizar uma granularidade semanal (com ciclo definido como falso) devido à pertinência de saber quais regras valem especificamente em cada dia da semana. Assim, independente da quantidade de semanas existente no arquivo de origem, apenas 7 iterações (uma para cada dia da semana) serão necessárias para a conclusão do algoritmo.

Mas se a necessidade for entender a mudança nos padrões de consumo, poderíamos adotar um ciclo com granularidade mensal, com n iterações (uma para cada mês existente no arquivo CSV).

Além destas, para organizar e simplificar a codificação, foram implementadas duas classes para ler e gravar dados no padrão CSV. A classe TCarregaCSV, é responsável por abrir o arquivo e carregar linha por linha com o intuito de construir o mapa de *itemsets* exigidos pela classe TApriori. A classe TSalvaCSV, é responsável em exportar as regras obtidas para um arquivo em disco também no formato CSV.

4.3. PREPARAÇÃO DOS DADOS

A estrutura do arquivo de entrada adotada consiste numa data de referência (no padrão YYYY-MM-DD) seguido dos códigos dos *itemsets* (produto, evento, etc.), obtendo desta forma várias linhas (onde cada uma representa uma transação) com o leiaute “data”; item 1; item 2;...; item n-1; item n;

Por definição do arquivo CSV, campos não-numéricos (datas, strings, etc.) possuem aspas delimitando seu início e término. Por isto a data que inicia a transação está entre *“aspas”*.

4.4. BASE DE DADOS ADOTADA

Todos os testes foram realizados sobre dados sintéticos. Foram criados alguns arquivos com o comportamento desejado e elaborado um gerador de cestas de supermercado baseado em conversa com alguns proprietários, sendo que um deles é responsável por dois supermercados com mais de 15 anos de atividade.

Como resultado, foi levantado o seguinte comportamento (assimilado ao gerador): (i) os sábados apresentam 70% mais vendas do que a média semanal e contém mais do que o dobro de produtos no mesmo ticket de vendas; (ii) às segundas-feiras ocorre uma pequena redução, tanto na média de vendas quanto na quantidade de itens no mesmo ticket; (iii) aos domingos há poucas transações, porém elas são muito maiores do que a média semanal de itens por ticket; (iv) os demais dias não apresentam diferença significativa entre a média de vendas ou de itens por ticket; (v) as vendas no mês de dezembro recebem um incremento substancial. Um comportamento cogitado, Mas que não foi implementado no gerador é a diferenciação de produtos supérfluos, que comumente são menos freqüentes no dia-a-dia, mas demonstram especial participação no final do ano e nos primeiros sábados do mês.

Desta forma, o gerador possui as seguintes variáveis: data inicial, quantidade de dias a gerar, tamanho da transação média, quantidade de transações diárias e quantidade de produtos. Sua sistemática acrescenta um desvio randômico na quantidade de transações diárias e no tamanho da transação média.

Para efeito demonstrativo, foram adotadas três diferentes bases de dados (superpostas entre si) com informações de dois meses (59 dias – 2.040 transações), um ano (365 dias – 13.407 transações) e quatro anos (1461 dias – 58.892 transações), simulando a existência de alguns produtos

que eram comercializados apenas no primeiro ano, e alguns que passaram a ser comercializados no final do quarto ano.

.

4.5. RESULTADOS OBTIDOS

Para efeito de comparação, foram realizadas as simulações com o percentual de suporte em 15%, 20% e 25%, exceto para o Apriori tradicional, exibidas na figura 6, cujas execuções adotaram suporte em 5%, 10%, e 15%, pois os testes com um suporte igual ou superior a 20% não encontraram regras.

Olhando detalhadamente as próximas 9 figuras (numeradas de 6 a 14), percebemos que ao aumentar o suporte (dentro da mesma massa de dados), a quantidade de regras encontradas é reduzida drasticamente. Como exemplo, na figura 6 para a base de 2 meses, passam de 1265 regras em 5% para apenas 10 regras com um suporte mínimo de 15%.

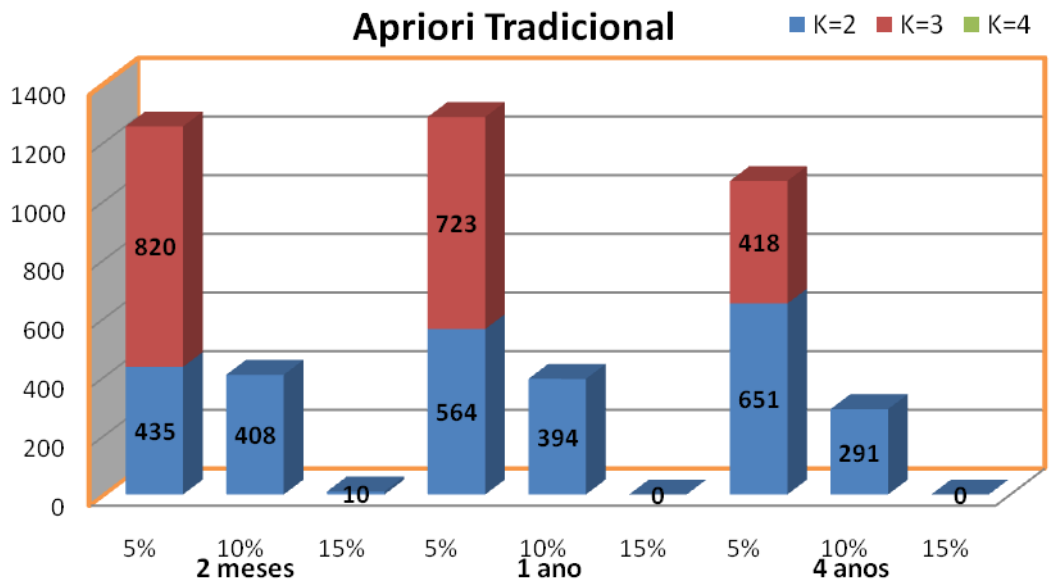


FIGURA 6 –QUANT. REGRAS ENCONTRADAS - APRIORI TRADICIONAL

Nas figuras 7, 8, 9 e 10, são apresentados as quantidades de regras encontradas pelo algoritmo proposto com a variável ciclo definida como

falso, fazendo com que a versão semanal (figura 7) apresente 7 iterações, a mensal (figura 8), 31 iterações (uma para cada dia do mês), a versão anual (figura 9) 12 iterações (uma por mês) e a semana do ano (figura 10) 54 iterações. Se comparadas com o Apriori tradicional, as quantidades de regras encontradas nas 3 bases foi muito maior pois, com 15% de suporte, o Apriori encontrou respectivamente 10, 0 e 0 regras.

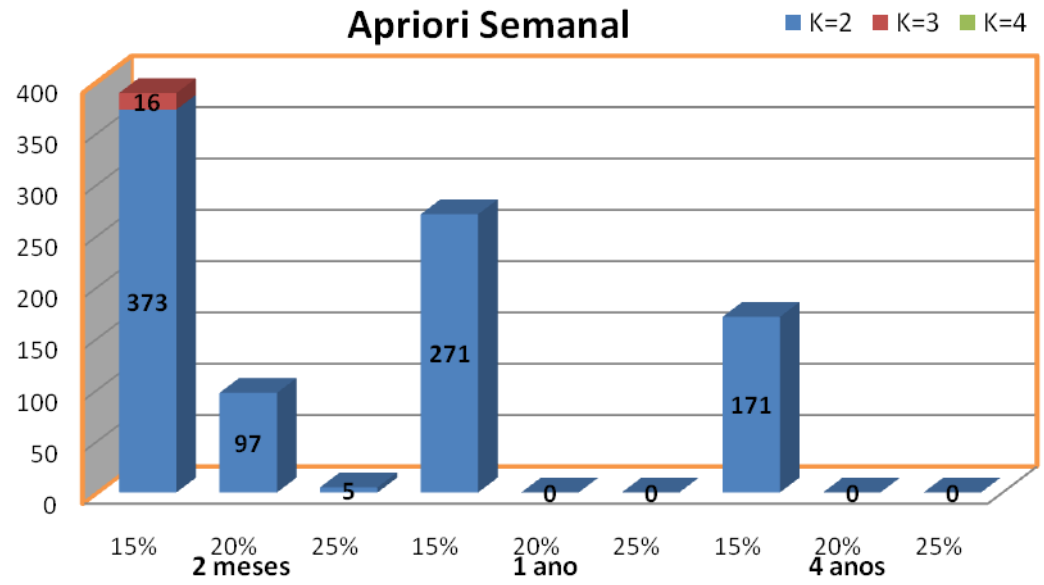


FIGURA 7 –QUANT. REGRAS ENCONTRADAS - APRIORI SEMANAL

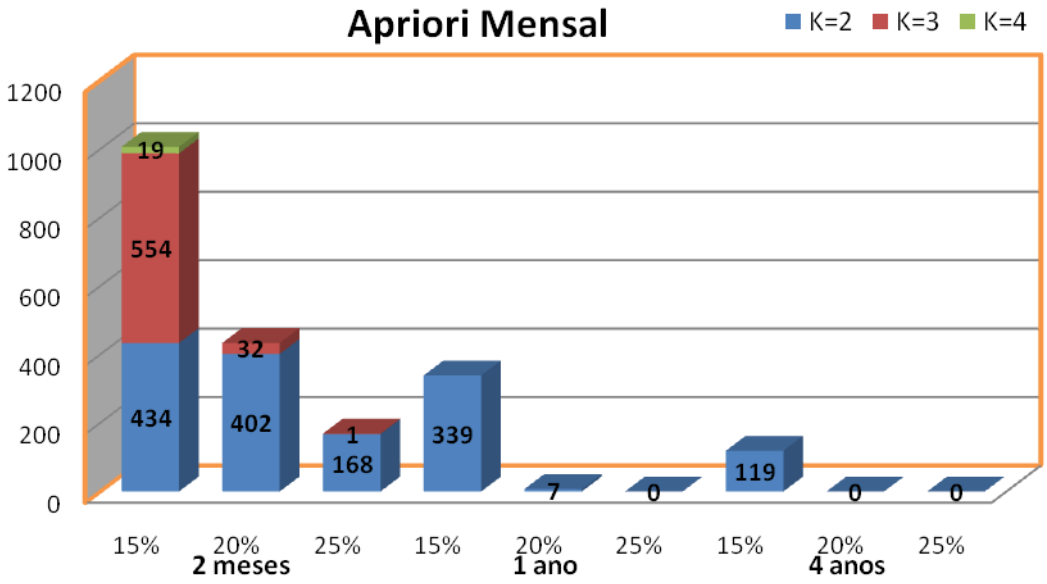


FIGURA 8 –QUANT. REGRAS ENCONTRADAS - APRIORI MENSAL

Este aumento da quantidade de regras com o mesmo percentual de suporte é compreensível, já que menos transações são utilizadas para descobrir cada regra. Com isto, regras expressivas que valham somente num determinado mês ou dia da semana passam a ser detectadas e sinalizadas, algo que dificilmente aconteceria na versão tradicional. Contudo, deve-se atentar que se a quantidade de iterações for muito alta, ou se for adotada uma granularidade muito fina, sob o risco de encontrar uma grande quantidade de regras que só sejam válidas apenas uma única vez no ano.

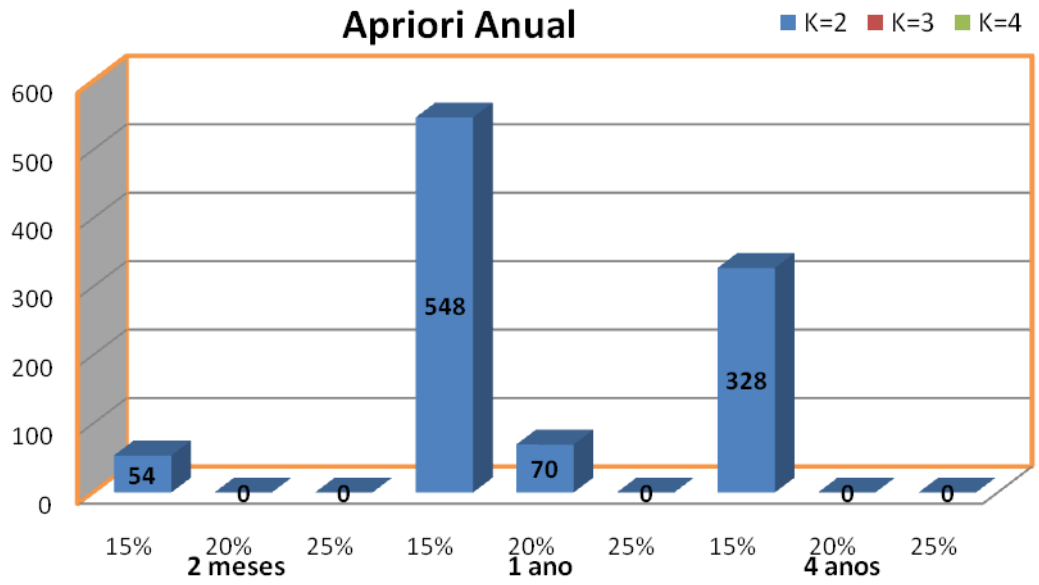


FIGURA 9 –QUANT. REGRAS ENCONTRADAS - APRIORI ANUAL

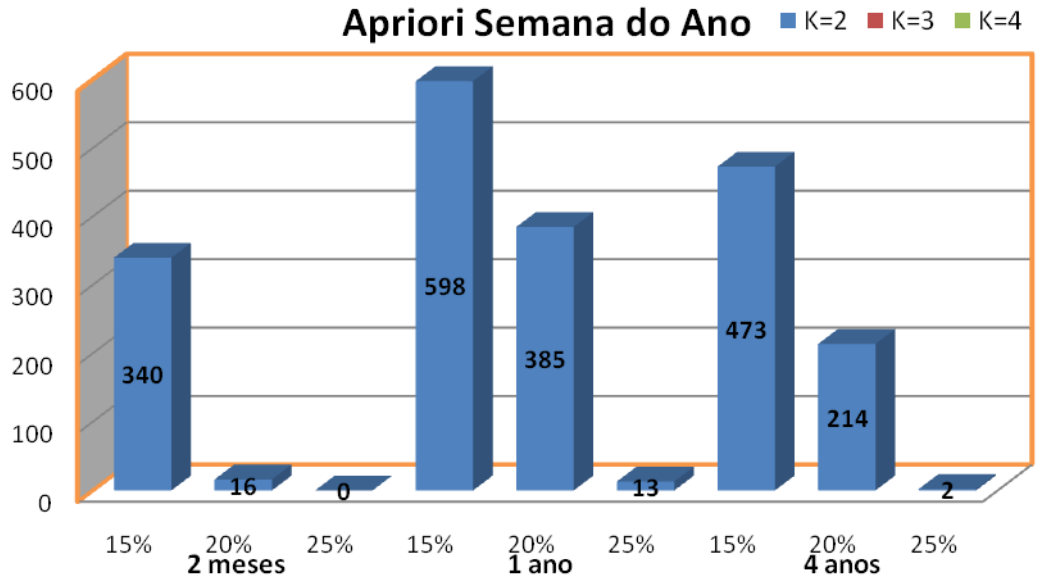


FIGURA 10 –QUANT. REGRAS ENCONTRADAS - APRIORI POR SEMANA DO ANO

Nas figuras 9 e 10, é facilmente percebida a baixa quantidade de regras encontradas na base de 2 meses. Isto é causado pelo volume muito pequeno de dados utilizados. Como é uma base de 60 dias, ou seja, dois meses, ou 9 semanas, um mapa gerado nas condições da figura 9, das 12 colunas apenas 2 seriam preenchidas, e na figura 10, para as 54 semanas que um ano possui, apenas 9 são valoradas.

Apesar dos 5 gráficos supracitados nesta seção demonstrarem que quanto maior a base (a base de 4 anos contém a base de um ano, que por sua vez contém a base de 2 meses) menos regras são encontradas, isto não é uma regra, haja visto que em alguns casos a base de um ano apresentou mais regras do que a base de 2 meses. Esta é uma base sintética que simula um comportamento específico onde alguns produtos foram comercializados apenas no início, enquanto outros foram comercializados apenas no final da mesma, e algumas regras têm maior probabilidade de ocorrer num determinado dia da semana. Mesmo que isto possa se refletir em boa parte dos estabelecimentos comerciais, seria necessário um amplo estudo, algo fora do escopo deste trabalho, para comprovar esta tendência.

Nas próximas 4 figuras (de 11 a 15) são exibidas as quantidades de regras encontradas com a variável ciclo definida como verdadeira.

Na figura 11, percebemos que um ciclo semanal (uma iteração a cada 7 dias) já pode ser considerada como baixa, pois mesmo uma taxa de suporte relativamente alta (25%) apresentou uma grande quantidade de regras encontradas. Apesar de pouco útil em seu estado bruto, o mapa gerado nesta granularidade possibilita detectar sazonalidades, mediante o uso de outras ferramentas para detecção de pico, amplitude, ponto-médio e distância entre os picos (ferramentas estatísticas por exemplo)

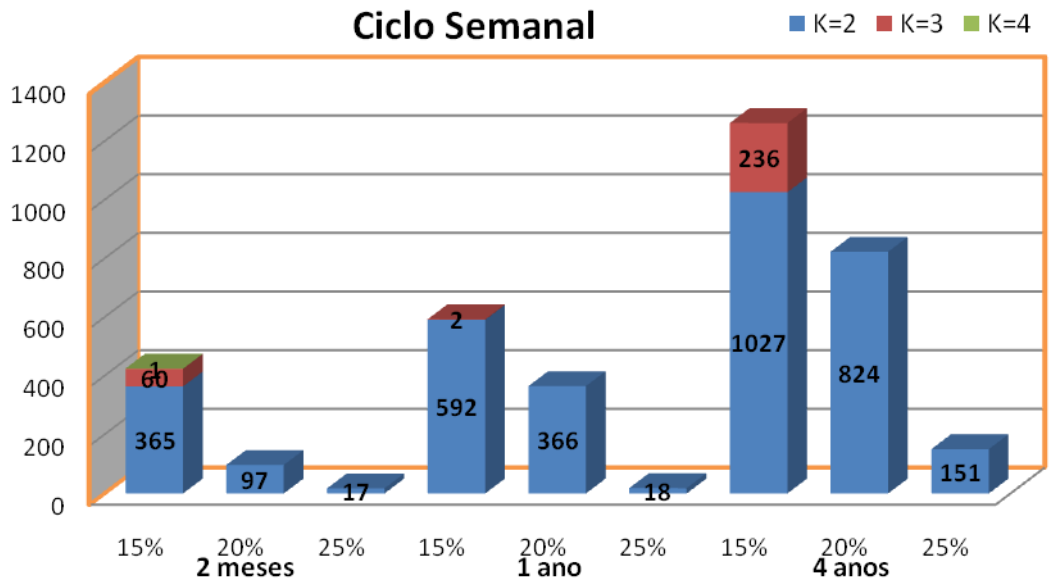


FIGURA 11 –QUANT. REGRAS ENCONTRADAS – CICLO SEMANAL

Com relação à figura 12, percebemos que as regras encontradas nas bases de 2 meses e 1 ano são exatamente as mesmas encontradas na figura 9 (Apriori anual) pois possuem a mesma regra e granularidade (intervalos de mês e mês), porém para a base de 4 anos, há 48 períodos na figura 12 contra apenas 12 exibidos na outra figura.

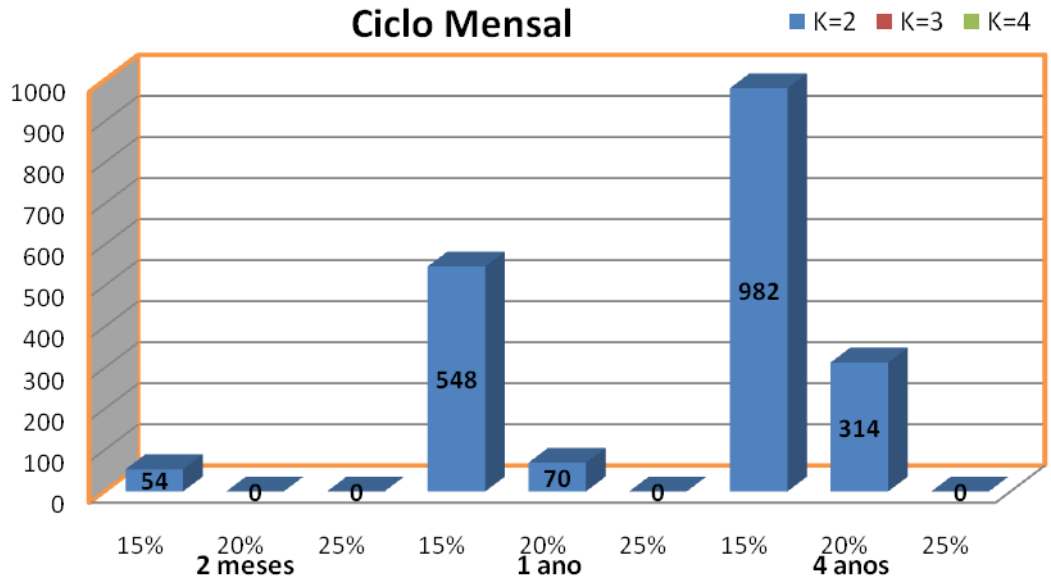


FIGURA 12 –QUANT. REGRAS ENCONTRADAS – CICLO MENSAL

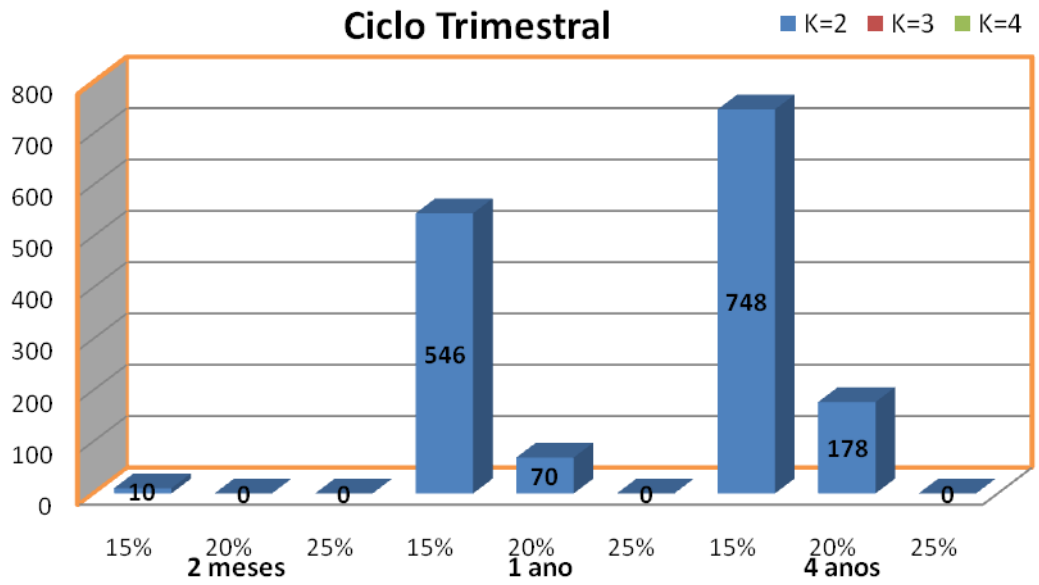


FIGURA 13 –QUANT. REGRAS ENCONTRADAS – CICLO TRIMESTRAL

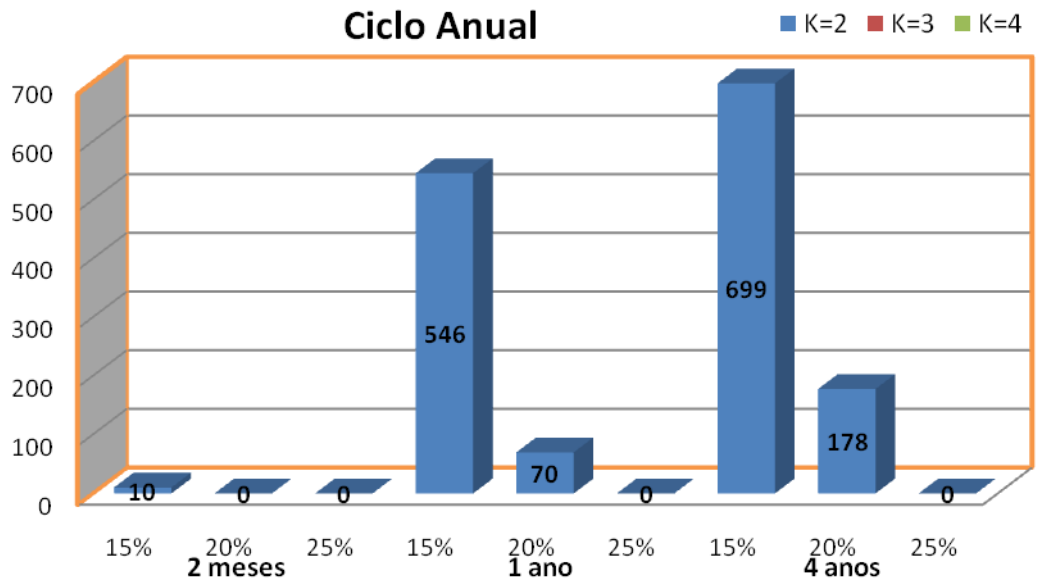


FIGURA 14 –QUANT. REGRAS ENCONTRADAS – CICLO ANUAL

Quando olhamos os 4 gráficos gerados com a variável ciclo definida como verdadeira, percebemos que com o aumento da base de dados, a quantidade de regras tende a ser igual ou maior, isto é normal pois, como dito anteriormente, quanto mais ciclos menos transações são analisadas por ciclo.

4.6. CONSIDERAÇÕES FINAIS

Diante dos dados obtidos através do algoritmo proposto, concluímos que há uma significativa mudança tanto na quantidade quanto na qualidade das regras resultantes, e que novos estudos se fazem necessários para ratificar certas tendências encontradas nas regras, enquanto que outras são mais fáceis de entender e explicar.

Apesar de aparentemente ser verdadeira, a redução da quantidade de regras encontradas pelo Apriori tradicional com o crescimento da base de dados, é algo que carece de mais estudos.

Porém não temos problemas em explicar a constatação que aponta para a qualidade das regras frente à quantidade de iterações, pois quanto mais iterações, menos transações são utilizadas na confecção de cada regra, o que pode causar um aumento na quantidade de regras encontradas com o mesmo suporte. Como conseqüência, boa parte destas regras tem uma validade extremamente restrita, ou seja, em apenas uma ou duas iterações temporais, e baixíssima utilidade prática. Tal comportamento pode ser corrigido acrescentando uma nova métrica – sugerida como trabalho futuro na seção 5.3 – chamada de **expressividade temporal**, sendo obtida pela quantidade de iterações em que a regra aparece dividida pelo total de iterações.

E apesar da base de dados ser sintética, constatamos que as suspeitas de que determinados padrões de consumo são válidos apenas numa janela temporal como um dia específico da semana (visto no exemplo da figura 5) ou mês específico do ano.

5. CONCLUSÕES

5.1. CONSIDERAÇÕES INICIAIS

Apesar de o algoritmo proposto ter uma implementação simples, a mesma exigiu muita pesquisa e experimentação. Ainda na fase de definição do tema, percebeu-se que há muitas formas diferentes, e pouco ortodoxas, para se trabalhar com o aspecto temporal. Algumas técnicas propostas são específicas para uma única realidade enquanto que outras aparentam ser simples provas de conceito. Quanto mais autores eram pesquisados, mais distante parecia a solução. A conclusão: trabalhar com o tempo exigirá pelo menos duas etapas.

Ainda no campo da experimentação, foram estudadas a viabilidade de fazer a amarração temporal após a obtenção das regras pelo Apriori, ou de realizar uma modificação no Apriori para que este traduzisse a base de dados numa lista de eventos temporais, onde cada evento sinalizasse a presença de um candidato numa janela temporal servindo de base para outra fase de mineração.

Tal abordagem demonstrou-se extremamente complexa, pouco produtiva, e incrivelmente devoradora de recursos como memória e processamento, pois tudo apontava para uma espécie de produto cartesiano entre cada transação com as demais transações da base a ser minerada em suas diferentes granularidades.

Após a percepção que é mais simples e fácil acrescentar a validade temporal numa regra já descoberta, foi avaliado a pertinência da solução no âmbito da mineração de dados em bases temporais.

5.2. CONTRIBUIÇÕES

Apesar de não ser um tema inédito, um algoritmo que descubra regras de associação, ou encontre a freqüência em que estas regras ocorram, acreditamos que o presente trabalho seja pertinente e apresente algumas contribuições tanto para usuários-finais quanto para a comunidade científica.

A primeira delas está na forma modular em que o algoritmo proposto faz o tratamento das regras e de sua ocorrência temporal, demonstrando ser flexível para atender não somente situações muito específicas, mas pode ser facilmente utilizado por virtualmente qualquer atividade que se beneficie da mineração por regras de associação como “tíquetes de venda” e “detecção de eventos”.

A segunda está na saída do algoritmo proposto: um mapa contendo as regras e sua validade temporal numa janela de tempo (sujeito às variáveis ciclo e granularidade), pois possibilita encontrar, mediante análise, características como sazonalidades, ocorrência e duração dos ciclos de repetição. E conseguir ainda mais informações através do uso de outras técnicas e ferramentas.

5.3. TRABALHOS FUTUROS

O algoritmo proposto, embora apresente algumas novidades e vantagens, ainda tem muito a evoluir. Uma das primeiras constatações de pontos passíveis de melhorias está na forma seqüencial adotada pelo mesmo, seria interessante aproveitar a onda de processadores de múltiplos núcleos e processar vários *threads* em paralelo. Dentro da mesma linha de raciocínio, ao invés de selecionar uma única granularidade (ou executar o algoritmo n vezes), acreditamos que este poderia ser mais inteligente e processar simultaneamente diferentes granularidades.

Outra questão interessante seria a adoção de uma equação estatística para calcular o ápice, ponto médio e duração de cada momento de validade das regras, permitindo encontrar uma função para estimar o ciclo de validade e repetição das regras com maior precisão.

Do ponto de vista temporal, poderíamos expandir a classe TTempo para que a mesma possa utilizar um “validador externo” visando ganhar a possibilidade de diferenciar e tratar corretamente feriados, festividades, dias úteis e dias letivos. Ainda nesta temática, seria interessante que o usuário pudesse informar a data inicial e final que quer minerar, desprezando partes do arquivo texto que estejam fora do escopo informado.

Apesar de ainda muito utilizado, o Apriori já perdeu o posto de algoritmo mais eficiente e performático faz algum tempo. Por esta razão acreditamos que se possam conseguir um excepcional ganho trocando-o pelo *FP-Growth*, entre outros.

Também constatamos que o presente algoritmo não traz um campo identificação (como código do cliente/fornecedor/aluno, etc.) o que possibilitaria detectar regras do tipo “A hoje -> B daqui aproximadamente X dias”.

Algo que ficou subentendido ao olhar os gráficos expostos na seção 4.5, é que ao utilizar um suporte muito baixo são encontradas muitas regras de pouca utilidade prática, e com um suporte muito alto há pouco sucesso em obter regras um pouco maiores. Isto demonstra que seria extremamente interessante incluir uma nova métrica (expressividade temporal) na etapa de consolidação dos dados.

5.4. CONCLUSÃO FINAL

Através de protótipos e experimentações, constatamos que deveríamos tratar a questão temporal de forma separada das regras, passando a filtrar / classificar os *itemsets* conforme sua concepção temporal antes da mineração com o Apriori, o que possibilitou encontrarmos facilmente as regras e consolidá-las com sua validade temporal, incorporando o Apriori praticamente intacto no interior do novo algoritmo.

A grande vantagem do algoritmo proposto é sua capacidade de demonstrar, não apenas a ocorrência, ou melhor, a existência de regras, mas também qual é a validade delas (freqüência), reflexo de sua identidade temporal, sem exigir muito trabalho do usuário.

Diante disto, acreditamos que o principal objetivo tenha sido atingido, e por estar longe de esgotar o assunto, certamente novas propostas de trabalho e pesquisa surgirão a partir do presente trabalho, produzindo um amplo leque de melhorias e possibilidades, dentre os quais algumas poucas foram explicitadas na seção 5.3, o que nos serve de estimulo e motivação.

REFERÊNCIAS BIBLIOGRÁFICAS

AGRAWAL Rakesh, IMIELINSKI Tomasz, SWAMI Arun. **Mining Association Rules Between Sets of Items in Large Database.** IBM Almaden Research Center, EUA, 1993.

ALE Juan M., ROSSI Gustavo H. **An Approach to Discovering Temporal Association Rules.** Proceedings of the 2000 ACM symposium on Applied computing – Volume 1**.** ACM, EUA, 2000.

BETTINI Claudio, JAJODIA Sushil, WANG Sean X. **Time Granularities in Databases, Data Mining and Temporal Reasoning.** Springer-Verlag New York, 2000

BETTINI Claudio, Dyreson Curtis E., Evans William S., Snodgrass Richard T. Wang Sean X. **A Glossary of Time Granularity Concepts.** Springer–Verlag Berlin Heidelberg, 1998.

BORGELT Christian.**An Implementation of the FPgrowth Algorithm**. Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations. ACM, Chicago, Illinois, 2005.

CASAS R., **Calendários**. Disponível em: <http://www.observatorio.ufmg.br/pas39.htm>. Acesso em: 04 mai 2009.

DATE C.J. **Introdução a Sistemas de Banco de Dados – Tradução da 8ª Edição Americana**. Campus, 2004.

DEVLIN B.A., MURPHY P.T. **An Architecture for a Business and Information System.** IBM Systems Journal 17, no. 1. Disponível em: <http://www.research.ibm.com/journal/sj/271/ibmsj2701G.pdf>. Acesso em: 20 nov 2007.

FAYYAD Usama M., PIATETSKY-SHAPIRO Gregory, SMYTH Padhraic, UTHURUSAMY Ramasamy (editores), **Advances in Knowledge Discovery and Mineração de Dados.** American Association for Artificial Intelligence, California – EUA, 1996.

FONG Joseph (editor). **Data Mining, Data Warehousing & Client/Server Databases – Proceedings of the 8th International Database Workshop.** Springer-Verlag. Singapura, 1997.

GRAAF Edgar de, KOSTERS Walter A. **Mining for Stable Patterns: Regular Interval Between Occurrences**. BNAIC – Bélgica 2006. Disponível em: http://www.liacs.nl/~edegraaf/papers/bnaic2006.pdf

HARMS Sherri K., DEOGUN Jitender, TADESSE Tsegaye. **Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences.** Foundations of Intelligent Systems. Springer Berlin, 2002.

HARTIGAN J.A., WONG M.A. **Algorithm AS 136: A K-Means Clustering Algorithm.** Applied Statistics, Vol 28, nº 1 - 1979.

INMON William .H., **Words of Wisdom from Bill Inmon**. Disponível em: <http://www.inmoncif.com/about/wordsofwisdom.php>. Acesso em: 20 nov. 2007.

INMON William H., WELCH J. D., GLASSEY Katherine L. **Gerenciando Data Warehouse.** Makron Books. São Paulo – SP, 1999.

JENSEN C.S., CLIFFORD J., GADIA S.K., SEGEV A. SNODGRASS R.T., **A Glossary of Temporal Database Concepts.** SIGMOD RECORD, Vol.21 Nº 3, 1992.

JENSEN C.S., CLIFFORD J., ELMASRI R., GADIA S.K., HAYES P., JAJODIA S. (editores). **A Consensun Glossary of Temporal Database Concepts.** SIGMOD RECORD Vol. 23 nº1, 1994.

KOSTERS, W.A., PIJLS W, POPOVA V. **Complexity Analysis of Depth First and FP-growth implementations of APRIORI.** Proceedings of MLDM 2003 (Machine Learning and Data Mining in Pattern Recognition), Springer-Verlag, Alemanha 2003.

LEVINE, David M., STEPHAN David, KREHBIEL Timothy C., BERENSON Mark L., **Estatística – Teoria e Aplicações Usando o Microsoft Excel em Português - 3ª Edição.** LTC, 2005.
LAXMAN Srivatsan, SASTRY P.S. **A Survey of Temporal Data Mining.** Springer India, Volume 31, Number 2 / April, 2006.

LI Yingjiu, WANG X. S., JAJODIA Sushil. **Discovering Temporal Patterns in Multiple Granularities.** Springer-Verlag, Alemanha, 2001

LI Zhi-Chao, HE Pi-Lian, LEI Ming. **A High Efficient AprioriTID Algorithm for Mining Association Rule.** Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

OLIVIERI Antonio C., **Calendário – História Geral – UOL Educação,** Disponível em: <http://educacao.uol.com.br/historia/ult1685u275.jhtm>. Acesso em: 04 mai 2009.

RAINSFORD Chris P., RODDICK John F. **Adding Temporal Semantics to Association Rules.** Springer-Verlag – Alemanha, 1999.

ROMÃO Wesley, NIEDERAUER Carlos A. P., MARTINS Alejandro, TCHOLAKIAN Aran, PACHECO Roberto C. S., BARCIA Ricardo. **Extração de regras de associação em C&T: o algoritmo Apriori.** XIX Encontro Nacional em Engenharia de Produção – Rio de Janeiro, 1999.

SHANG Xuequn, SATTLER Kai Uwe. **Depth-First Frequent itemset Mining in Relational Databases.** (Proceedings of the 2005 Symposium on Applied Computing). ACM, Santa Fe, New Mexico, 2005

TAMIR Raz, YEHUDA Singer. **On a confidence gain measure for association rule discovery and scoring.** Springer-Verlag, 2006

WHEELWRIGHT, Steven C.; MAKRIDAKIS, Spyros. **Forecasting Methods for Management. 4th edition**. New York : John Wiley & Sons Inc, 1985.