

**UNIVERSIDADE ESTADUAL PAULISTA**  
**"JÚLIO DE MESQUITA FILHO"**  
Faculdade de Ciências - Campus Bauru

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**SOBRE REGRAS DE ASSOCIAÇÃO  
UTILIZANDO MINERAÇÃO DE DADOS**

BAURU  
2016

GUSTAVO HENRIQUE DE ROSA

## **SOBRE REGRAS DE ASSOCIAÇÃO UTILIZANDO MINERAÇÃO DE DADOS**

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.

Orientador: Prof. Dr. João Paulo Papa

Universidade Estadual Paulista “Júlio de Mesquita Filho”

Faculdade de Ciências

Ciência da Computação

BAURU  
2016

Gustavo Henrique de Rosa

Sobre Regras de Associação utilizando Mineração de Dados/ Gustavo Henrique de Rosa. – Bauru, 2016-

94 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. João Paulo Papa

Trabalho de Conclusão de Curso – Universidade Estadual Paulista “Júlio de Mesquita Filho”  
Faculdade de Ciências  
Ciência da Computação, 2016.

1. Inteligência Artificial 2. Big Data 3. Análises Estatísticas 4. Problemas de Mercado 5. Mineração de Dados 6. Regras de Associação 7. Suporte e Confiança 7. Conjuntos de Itens 8. Conjuntos Candidatos e Frequentes 9. Metodologia *Apriori* 10. FP-Growth

Gustavo Henrique de Rosa

## **Sobre Regras de Associação utilizando Mineração de Dados**

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.

Banca Examinadora

---

**Prof. Dr. João Paulo Papa**  
Orientador

---

**Prof. Dr. Aparecido Nilceu Marana**

---

**Prof. Dr. Kelton Augusto Pontara da Costa**

Bauru, \_\_\_\_\_ de \_\_\_\_\_ de \_\_\_\_\_.

*Este trabalho é dedicado à humanidade que,  
quando for unida, será capaz de se tornar uma grande ciência.*

# Agradecimentos

Em primeiro lugar, gostaria de agradecer à fé, que independente da religião, doutrina, ensinamentos, deve sempre existir e prevalecer, principalmente a fé no amor e na vida, os quais são sem dúvida os grandes motivos de estarmos presentes nessa passagem.

Agradeço aos meus pais, Ana Valéria e Fernando, por sempre estarem presentes em minha vida, me auxiliando e fortalecendo nos momentos que mais precisei, por sempre apoiarem as minhas decisões e me incentivarem em tudo que faço.

Agradeço à minha família, à minha prima Bárbara, aos meus tios Fabrício e Helena, Luciene e Marcos, Helen, Daiane e João, por todas as nossas conversas, ensinamentos, tudo que eles puderam proporcionar e contribuir à minha formação. Agradeço em especial aos meus avós, José e Thereza, Lurdinha e principalmente ao meu falecido vô Mauro, pessoa mais nobre e batalhadora que já conheci e que infelizmente só pude entender tardeamente o quanto grande e evoluído era.

Agradeço à minha pequena, Bruna, a qual eu sempre amarei mais que tudo e que toda noite agradeço por ter aparecido em minha vida. Sem dúvida todo o tempo que passamos juntos e que iremos passar me faz ser uma pessoa melhor e me motivar mais para, juntos, melhorarmos o mundo e crescemos. Toda a sua paciência e amor que tem comigo é algo inacreditável, o qual espero poder retribuir todos os dias da mesma forma.

Agradeço ao meu orientador, João Paulo Papa, o qual sempre foi uma fonte de inspiração para mim, seguir os seus passos é somente umas das coisas que mais almejo, podendo ter certeza de que ele é e sempre será um dos meus maiores orgulhos.

Agradeço à todos os meus amigos de Brasília, principalmente os de infância e que estiverem sempre comigo, independente da distância. Ao grupo Brothers, que hoje em dia, nos reúne e acaba por fortalecer os nossos laços. Em especial à Rafaela, a qual conheço somente há 21 anos e que independente da distância, sempre que nos reencontramos acaba sendo o mesmo amor e amizade de sempre.

Agradeço aos meus amigos de Bauru, especialmente àqueles que puderam estar comigo durante esses quatro anos, à nossa falecida república Alabama, local dos melhores encontros. Pala, Torta, Plínio, Flávio, Siri, Grecin, Ceja, Dedê, Gabriel, Gui, Elisa. Agradeço também aos meus bixos diretos, agregados e veteranos, especialmente aos da Toca e do Pérola Negra, por todos os ensinamentos passados. Agradeço à nossa querida organização da Psicomp, em especial à Ana, Gabi, Raíssa e Renata, por toda a amizade e ajuda nesse período de graduação. Agradeço também a todos os outros amigos de outros cursos e pessoas que estiveram presentes em minha vida, Stéphanie, Fernanda, Denise, Nathalie, Dine, Bruna, Bárbara, Camila.

Agradeço à Naumteria, a melhor bateria do mundo, a qual pude participar em meu último ano. O sentimento de ser unespiano e poder vestir as cores do nosso campus todas as semanas, ensaiando debaixo do sol e por um motivo tão nobre como a união e a música, são fatos indescritíveis e que levarei comigo para toda a vida.

Agradeço à Jr.COM, nossa empresa júnior de computação, local de grandes estudos e aprendizados, os quais tenho certeza de que ajudarão em toda minha vida.

Agradeço à UNESP Bauru, a única universidade a qual gostaria de ter participado e a qual é o meu maior orgulho de poder fazer parte e ser considerado um unespiano. Agradeço também à todos os professores do DCo e da FC, em especial ao Humberto, Nilceu e Kelton, os quais contribuíram enormemente para a minha formação acadêmica e social.

*"Emancipate yourselves from mental slavery.  
None but ourselves can free our minds."  
(Redemption Song, Bob Marley)*

# Resumo

A tecnologia vem proporcionando enormes avanços em nossas atividades do cotidiano. Atualmente, o mundo capitalista exige novos processos e tendências a fim de solucionar seus problemas de competitividade e aprimorar sua metodologia de operação. A área da mineração de dados tornou-se extremamente responsável por auxiliar na tomada de decisão das empresas, onde a efetiva análise dos dados armazenados resultantes das próprias operações da empresa conseguiram proporcionar conhecimentos de alto nível e importantes associações. Entretanto, ao passo em que a tecnologia se aprofunda e desenvolve cada vez mais, a quantidade de dados e objetos conectados à essa rede também cresce exponencialmente, sendo necessárias novas abordagens e metodologias capazes de aprimorar e otimizar os processos já utilizados. Dentro desse panorama, as chamadas regras de associação constituem uma importante abordagem para a descoberta dessas informações de alto nível. Representada por um modelo formal matemático e sustentada por diversos teoremas, essa subárea da mineração de dados é capaz de produzir ótimos resultados, principalmente para problemas associados ao mercado, sendo esse garimpo de regras, uma importante fonte comercial para a empresa utilizadora. A utilização de dois importantes algoritmos para a mineração de dados, *Apriori* e FP-Growth, fornece um maior entendimento sobre a geração de regras de associação, ao passo que também é de grande necessidade testar, validar e comparar essas duas técnicas com bases de dados reais e sintéticas. Os experimentos realizados comprovam a dificuldade na análise em grandes quantidades de dados devido ao crescimento exponencial da complexidade do problema, mostrando que também é necessária uma análise qualitativa das regras geradas, e não somente quantitativa.

**Palavras-chave:** Big data. Competitividade. Mineração de dados. Regras de associação. Otimização.

# Abstract

Nowadays, several of our daily activities are getting enhanced by technology. One may observe the capitalist world demands new processes and trends in order to solve its competitiveness problems, trying to improve its own methodology of operation. In light of, that data mining area has become extremely responsible for helping companies on decision-making tasks, where an effective analysis from the company data is used to provide high-level knowledges and important associations. However, whereas technology gets increasingly deep and more developed, the amount of data and objects connected to the network also grow exponentially, being necessary new approaches and methodologies that are able to improve and optimize the current procedures. Regarding this background, the well-known association rules compose an important approach for finding these high-level informations. Represented by a mathematical model and supported by several theorems, this data mining subarea is able to create excellent results, mainly for market-related problems, being this rule mining an important commercial source for the user undertaking. As briefly said, the need for better and more efficient algorithms ended up motivating the research on this area. The use of two important algorithms for data mining, Apriori and FP-Growth, provides a better comprehension about the association rules generation, whereas is still important to test, validate and compare these techniques with real and synthetic datasets. The experiments conducted justify the toughness of analyzing great amounts of data, mainly due to the problem complexity exponential growth, showing that is also necessary a qualitative analysis of the created rules, and not only quantitative.

**Keywords:** Big data. Competitiveness. Data Mining. Association Rules. Optimization.

# Listas de ilustrações

Figura 1 – Disposição de produtos em uma prateleira de supermercado.	16
Figura 2 – Fluxo de dados do funcionamento de uma atividade de clusterização e posterior treinamento em uma rede neural.	17
Figura 3 – Gráfico para exemplificação da atividade de uma técnica de agrupamento.	18
Figura 4 – Gráfico da quantidade de transistores ao longo do tempo de acordo com a Lei de Moore.	19
Figura 5 – A conexão de dispositivos na rede chamada “Internet das Coisas”.	21
Figura 6 – Ilustração de um simples processo de mineração em uma base de dados.	23
Figura 7 – Estrutura em grade de um conjunto de itens.	30
Figura 8 – Contagem do suporte de conjunto de itens candidatos.	31
Figura 9 – Ilustração do princípio <i>Apriori</i> . Caso $\{c, d, e\}$ seja frequente, todos seus subconjuntos serão frequentes.	32
Figura 10 – Ilustração do corte baseado no suporte. Caso $\{a, b\}$ seja infrequente, todos seus superconjuntos serão infrequentes.	33
Figura 11 – Ilustração da geração de conjuntos de itens frequentes pelo algoritmo <i>Apriori</i> .	34
Figura 12 – Geração e corte de conjuntos de $k$ itens candidatos através do método $F_{k-1} \times F_1$ .	38
Figura 13 – Geração e corte de conjuntos de $k$ itens candidatos através do método $F_{k-1} \times F_{k-1}$ .	39
Figura 14 – Geração e corte de conjuntos de $k$ itens candidatos através do método de força bruta.	39
Figura 15 – Enumeração de subconjuntos de três itens de uma transação $t$ .	40
Figura 16 – Ilustração da estrutura de uma árvore de <i>hash</i> .	41
Figura 17 – Conjuntos de itens frequentes máximos.	43
Figura 18 – Conjuntos de itens frequentes fechados com suporte mínimo de 40%.	45
Figura 19 – Diagrama de Venn dos conjuntos de itens frequentes.	46
Figura 20 – Construção de uma árvore FP.	47
Figura 21 – Construção de uma árvore FP utilizando um diferente método de ordenação do suporte de seus itens.	48
Figura 22 – Aplicação do algoritmo <i>FP-growth</i> para encontrar os conjuntos de itens frequentes terminados em $e$ .	50
Figura 23 – Diferentes tipos de busca para procura de conjuntos de itens frequentes.	52
Figura 24 – Tipos de árvores utilizadas na metodologia das classes equivalentes.	53
Figura 25 – Exemplos de busca em profundidade e largura.	54
Figura 26 – Exemplos dos possíveis layouts de estrutura para armazenamento dos dados.	55
Figura 27 – Processo de mineração do algoritmo AIS.	59
Figura 28 – Algoritmo AprioriTid.	60
Figura 29 – Corte de regras de associação baseado na métrica de confiança.	63

Figura 30 – Análise de compras de um supermercado.	66
Figura 31 – Escopo das categorias utilizadas como sub-bases.	68
Figura 32 – Detalhamento dos experimentos realizados.	73
Figura 33 – Gráfico do suporte mínimo x quantidade de conjuntos para a base de dados real.	74
Figura 34 – Gráfico do suporte mínimo x quantidade de conjuntos para a sub-base de dados informática.	75
Figura 35 – Gráfico do suporte mínimo x quantidade de conjuntos para a sub-base de dados conveniência.	76
Figura 36 – Gráfico do suporte mínimo x quantidade de regras para a base de dados real (confiança mínima 80%).	76
Figura 37 – Gráfico do suporte mínimo x quantidade de regras para a base de dados real (confiança mínima 90%).	77
Figura 38 – Gráfico do suporte mínimo x quantidade de regras para a base de dados real (confiança mínima 100%).	77
Figura 39 – Gráfico da confiança mínima x quantidade de regras para a base de dados real (suporte mínimo 0.0005%).	78
Figura 40 – Gráfico da confiança mínima x quantidade de regras para a base de dados real (suporte mínimo 0.001%).	78
Figura 41 – Gráfico do suporte mínimo x quantidade de conjuntos para a base de dados sintética 1K.	79
Figura 42 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 1K (confiança mínima 80%).	80
Figura 43 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 1K (confiança mínima 90%).	80
Figura 44 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 1K (confiança mínima 100%).	81
Figura 45 – Gráfico da confiança mínima x quantidade de regras para a base de dados sintética 1K (suporte mínimo 1%).	81
Figura 46 – Gráfico do suporte mínimo x quantidade de conjuntos para a base de dados sintética 10K.	82
Figura 47 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 10K (confiança mínima 80%).	82
Figura 48 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 10K (confiança mínima 90%).	83
Figura 49 – Gráfico da confiança mínima x quantidade de regras para a base de dados sintética 10K (suporte mínimo 0.35%).	83
Figura 50 – Gráfico do suporte mínimo x tempo de execução para a base de dados real (confiança mínima 80%).	84

Figura 51 – Gráfico da confiança mínima x tempo de execução para a base de dados real (suporte mínimo 0.0005%).	84
Figura 52 – Gráfico do suporte mínimo x tempo de execução para a base de dados sintética 1K (confiança mínima 90%).	85
Figura 53 – Gráfico da confiança mínima x tempo de execução para a base de dados sintética 1K (suporte mínimo 1%).	85
Figura 54 – Gráfico do suporte mínimo x tempo de execução para a base de dados sintética 10K (confiança mínima 80%).	86
Figura 55 – Gráfico da confiança mínima x tempo de execução para a base de dados sintética 10K (suporte mínimo 0.35%).	86

# Listas de tabelas

Tabela 1 – Exemplo de transações de mercado.	26
Tabela 2 – Representação binária das transações de mercado.	27
Tabela 3 – Lista dos conjuntos de itens frequentes correspondentes aos seus sufixos.	49
Tabela 4 – Exemplo de uma tupla da base de dados original.	67
Tabela 5 – Exemplo de uma transação da base de dados remodelada.	68
Tabela 6 – Exemplo de uma tupla da tabela ‘usuarios’ da base sintética.	69
Tabela 7 – Exemplo de uma tupla da tabela ‘canais’ da base sintética.	70
Tabela 8 – Exemplo de uma tupla da tabela ‘canais_usuarios’ da base sintética.	70
Tabela 9 – Exemplo de uma transação da base de dados sintética remodelada.	70
Tabela 10 – Distribuição da porcentagem de pacotes empregada na geração dos dados.	71
Tabela 11 – Distribuição de canais por pacote.	71
Tabela 12 – Regras com maiores valores de suporte e confiança para a base de dados real.	87
Tabela 13 – Regras com maiores valores de suporte e confiança para a base de dados sintética 1K.	88
Tabela 14 – Regras com maiores valores de suporte e confiança para a base de dados sintética 10K.	88

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>16</b>
<b>1.1</b>	<b>Modelagem do Problema</b>	<b>19</b>
1.1.1	O problema	19
<b>1.2</b>	<b>Objetivos</b>	<b>20</b>
1.2.1	Objetivo geral	20
1.2.2	Objetivos específicos	20
1.2.3	Justificativa	21
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>23</b>
<b>2.1</b>	<b>Conceitos em Mineração de Dados</b>	<b>23</b>
2.1.1	Classificação	24
2.1.2	Associações	24
2.1.3	Sequências	25
<b>2.2</b>	<b>Regras de Associação</b>	<b>25</b>
2.2.1	Modelo formal	26
2.2.2	Geração de conjuntos de itens frequentes	30
2.2.3	Representações de conjuntos de itens frequentes	42
2.2.4	Algoritmo <i>FP-growth</i>	46
2.2.5	Métodos alternativos para geração de conjuntos de itens frequentes	51
2.2.6	Outros algoritmos geradores	55
2.2.7	Geração de regras	62
<b>3</b>	<b>DESENVOLVIMENTO</b>	<b>65</b>
<b>3.1</b>	<b>Método de Pesquisa</b>	<b>65</b>
<b>3.2</b>	<b>Bases de Dados</b>	<b>66</b>
3.2.1	Abordagem do problema real	66
3.2.2	Abordagem do problema sintético	69
<b>3.3</b>	<b>Algoritmos Utilizados</b>	<b>72</b>
3.3.1	<i>Apriori</i>	72
3.3.2	<i>FP-growth</i>	72
<b>3.4</b>	<b>Experimentos</b>	<b>73</b>
<b>3.5</b>	<b>Resultados</b>	<b>74</b>
3.5.1	Base real	74
3.5.2	Base sintética 1K	79
3.5.3	Base sintética 10K	82
3.5.4	Tempo de execução	84

<b>3.6</b>	<b>Regras Geradas</b>	<b>87</b>
<b>4</b>	<b>CONCLUSÃO</b>	<b>90</b>
	<b>REFERÊNCIAS</b>	<b>92</b>

# 1 Introdução

A forte concorrência do capitalismo tem implicado constantemente em avanços no estudo de técnicas responsáveis por auxiliar tarefas de tomada de decisão (STONEBRAKER et al., 1993), dado que estratégias de mercado mal posicionadas ou até mesmo pequenos erros podem acarretar uma série de prejuízos para a sua executora. O rápido desenvolvimento da tecnologia proporcionou mudanças significativas em suas abordagens, possibilitando efetuar e armazenar milhões de cálculos em poucos segundos. Contudo, essa forte interação computacional entre a sociedade também é responsável por gerar um enorme fluxo de dados, os quais muitas vezes encontram-se dispersos e sem quaisquer regras de associações entre si.

O leitor pode se deparar com um grande número de atividades do cotidiano que poderiam ser armazenadas no formato de dados, para posterior aplicação de técnicas criadoras de regras de associação e classificação. Por exemplo, ao irmos ao supermercado acabamos comprando determinados objetos simplesmente por sua localização nas prateleiras ou até mesmo porque estamos inseridos em perfis específicos de consumidores, como estudantes, por exemplo, os quais normalmente compram cerveja e macarrão instantâneo.

Figura 1 – Disposição de produtos em uma prateleira de supermercado.



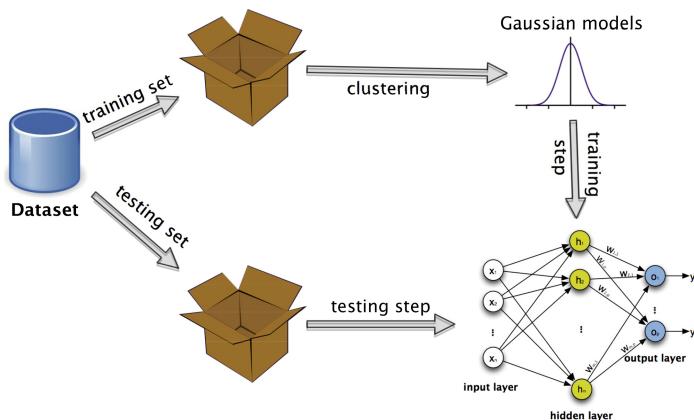
Fonte: Elaborada pelo autor.

Como mostrado pela Figura 1, é de extrema importância a existência de uma lógica na disposição física de produtos dentro de uma prateleira, tanto para o exemplo abordado pela imagem como para problemas em outros tipos de estabelecimentos, desde pequenas lojas à grandes revendedores. Entretanto, muitas vezes não é possível estabelecer o melhor padrão para a resolução do mesmo, sendo necessária uma análise mais detalhista e completa da situação.

Essa necessidade de analisar dados e transformá-los em possíveis informações úteis, bem como capazes de gerar futuros conhecimentos quando associadas, motivou o amplo estudo na área de Mineração de Dados ([AGRAWAL; IMIELINSKI; SWAMI, 1993a](#)). Esta técnica é baseada no aprendizado de máquina e, ao invés de classificar os dados, é responsável por encontrar padrões “interessantes” entre os mesmos, puramente motivado por interesses comerciais.

Três distintas classes de abordagens descritivas de mineração de dados (associações, sequências e classificações) foram propostas ([AGRAWAL; IMIELINSKI; SWAMI, 1993b](#)) e introduzidas, com a argumentação de que estas problemáticas poderiam ser uniformemente vistas como uma simples descoberta de regras embutidas em extensivas quantidades de dados. Para cada nova regra descoberta, alguns parâmetros estatísticos também são utilizados para aprimorar sua eficiência como, por exemplo, a confiança e o suporte da regra.

Figura 2 – Fluxo de dados do funcionamento de uma atividade de clusterização e posterior treinamento em uma rede neural.

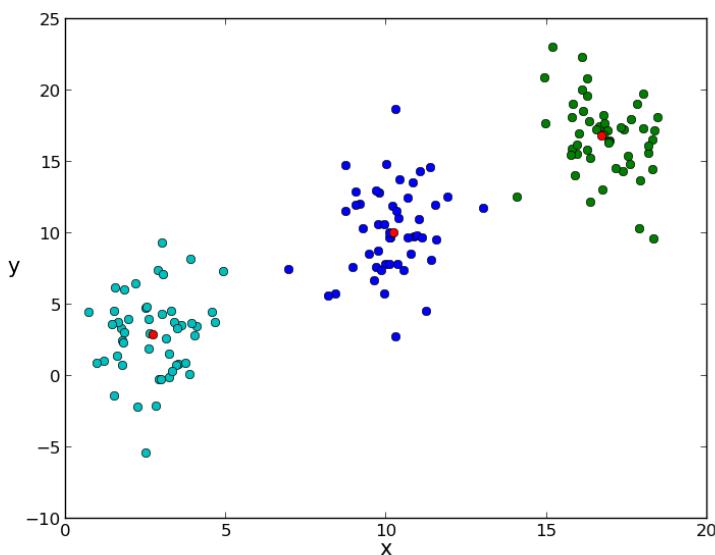


Fonte: Elaborada pelo autor.

Um dos principais problemas relacionados ao uso dessas técnicas consiste em armazenar grandes quantidades de dados ao longo do tempo para uso futuro, fato que dificulta sua aplicação em problemas de tempo real ([AGRAWAL; PSAILA, 1995](#)) ou até mesmo contribui para a criação de regras duplicadas ou ineficientes. Concomitantemente, o uso de técnicas para etapas de pré-processamento antes da aplicação de algoritmos de mineração tornou possível o corte e a diminuição de informações dentro desses grupos de dados. Um bom exemplo seria a utilização de ferramentas de agrupamentos de dados, do inglês *clustering*, atividade ilustrada pela Figura 2.

Essa técnica é responsável por elaborar e agrupar uma coleção de objetos de dados, fornecendo regiões de similaridade internas aos objetos do *cluster* e dissimilaridade para objetos em outros *clusters*. Como sua classificação é não-supervisionada, ou seja, não existe uma pré-criação de classes ou pré-entendimento sobre os dados, o processo de agrupamento é capaz de agrupar os dados a partir de um conjunto de regras definidas, proporcionando valiosas informações para a sua futura análise (HAN; KAMBER, 2000).

Figura 3 – Gráfico para exemplificação da atividade de uma técnica de agrupamento.



Fonte: Elaborada pelo autor.

A Figura 3 mostra um exemplo de gráfico gerado após a atividade de agrupamento em um espaço de dados. Os pontos coloridos correspondem às amostras de uma determinada classe de dados envolvida no exemplo, sendo que o ponto vermelho de cada distribuição é o centro de massa de cada agrupamento dos dados, tornando os dados inerentes ao agrupamento responsáveis por representar a classe rotulada.

Em contrapartida, essa atividade é altamente custosa para um elevado número de dados, exigindo processos de otimização a fim de aprimorar os seus resultados. Outra técnica possível para se minerar dados são as chamadas regras de associação, de caráter mais simples e efetivo para a resolução do problema apresentado por este projeto.

Portanto, este trabalho propõe a aplicação de diferentes técnicas de análise e processamento de dados com o intuito de apresentar as existentes abordagens para o problema em questão, estabelecendo um melhor entendimento sobre esta nova área, a qual visa prever e orientar o mercado para a produção do que o consumidor mais deseja, aprimorando o atual modelo de mercado empregado pelas empresas. O restante deste documento está organizado como segue. A Seção 1.1.1 apresenta a problematização da pesquisa. A Seção 1.2.3 apresenta

as justificativas acerca da resolução desse problema e do seu ganho para a comunidade científica. As Seções 1.2 e 3.1 descrevem os objetivos e a metodologia proposta para a resolução da abordagem apresentada, respectivamente. A fundamentação teórica está embasada e apresentada na Seção 2. O desenvolvimento e os resultados experimentais são discutidos na Seção 3. Finalmente, a conclusão está descrita na Seção 4.

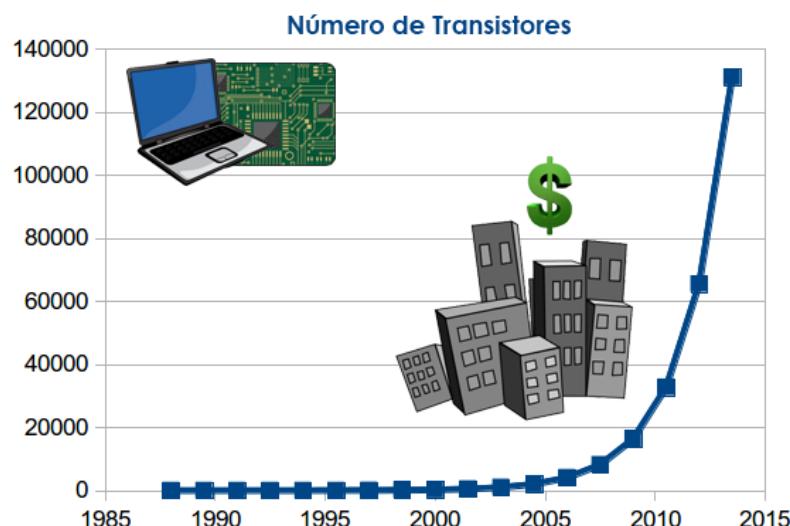
## 1.1 Modelagem do Problema

### 1.1.1 O problema

A mineração de dados sustenta-se com base em outras áreas da Ciência da Computação, sendo representada por um aglomerado de diferentes ferramentas capazes de serem utilizadas em conjunto, propiciando resultados mais confiáveis e robustos. Ideias advindas do aprendizado de máquina, reconhecimento de padrões, banco de dados e até mesmo da estatística são responsáveis por transformar a mineração de dados em uma tecnologia capaz de encontrar padrões de informações “escondidos” em determinados espaços.

Entretanto, é possível afirmar que enormes volumes de dados estão sendo coletados e armazenados, dado que os computadores, regidos pela Lei de Moore ([MOORE, 1965](#)), ilustrado pela Figura 4, também tiveram seu processamento ampliado em um ritmo exponencial. O aumento da pressão competitiva entre as empresas também fomenta o aprimoramento dessas técnicas, permitindo, do ponto de vista comercial, a criação de informações valiosas que serão utilizadas no auxílio do processo de tomada de decisão, contribuindo para o crescimento da empresa.

Figura 4 – Gráfico da quantidade de transistores ao longo do tempo de acordo com a Lei de Moore.



Fonte: Elaborada pelo autor. Dados provenientes de ([MOORE, 1965](#)).

A efetiva análise de dados de uma empresa pode resultar em associações e pesquisas de mercado muito qualificadas, tornando-se uma ótima opção para enfrentar um simples ou complexo problema de tomada de decisão. Contudo, a computação ainda demanda uma certa complexidade nas atividades em que efetua. O grande volume de dados pode ser visto como ouro aos olhos de uma empresa, mas muitas vezes este ouro está tão escondido e quebrado que são necessárias formas de recuperá-lo e colocá-lo em um formato compreensível.

O presente projeto almeja comparar algumas das técnicas existentes de modelagem e descrição de padrões da mineração de dados para a geração de regras de associação, principalmente àquelas contidas no modelo descritivo, o qual é o modelo responsável por identificar padrões ou relacionamentos entre os dados, de origem histórica ou não.

A utilização de ferramentas de regras de associação são vistas como o foco maior deste projeto, também acompanhadas de análises estatísticas e matemáticas, proporcionando um aprimoramento dos conhecimentos nas diferentes áreas que compõem a mineração de dados e aumentando a confiabilidade dos resultados obtidos.

## 1.2 Objetivos

### 1.2.1 Objetivo geral

Analisar as técnicas existentes na área de mineração de dados utilizando bases de dados reais e sintéticas para a validação e comparação das técnicas, com o intuito de criar um novo material acerca das ferramentas e abordagens já conhecidas e utilizadas por essa área, só que aplicadas dentro do tema principal deste projeto, isto é, um estudo mais aprimorado sobre as regras de associação.

### 1.2.2 Objetivos específicos

Atualmente, existe um paradigma relacionado à análise de grandes quantidades de dados e ao método de abordagem para se efetuar esta análise, dado que são necessários modelos menos complexos e que executem sua tarefa principal com uma alta eficiência ([AGRAWAL; SRIKANT, 1994](#)).

A validação e comparação de soluções existentes é de grande valia para este trabalho, sendo os passos necessários à sua obtenção descritos a seguir:

- a) levantar informações sobre o que é a mineração de dados, como ela é feita e quais são as principais abordagens em uso;
- b) focar principalmente na parte de regras de associações e estudar sobre alguns algoritmos nesta subárea;

- c) abordar uma aplicação viável a partir desse estudo inicial e encaixá-la no problema descrito por este projeto;
- d) aplicar as existentes técnicas no problema inicial contemplado por este projeto, para posterior validação em bases de dados reais e sintéticas;
- e) analisar o problema real para levantamento de informações para a construção de sua base de dados;
- f) criar um algoritmo de geração dos dados do problema da base de dados sintética;
- g) testar e validar as bases de dados real e sintética obtidas; e
- h) verificar os resultados obtidos entre as técnicas utilizadas através de métodos estatísticos e matemáticos, promovendo uma maior confiabilidade à pesquisa.

### 1.2.3 Justificativa

O avanço da tecnologia vem proporcionando uma maior conectividade entre os mais diversos dispositivos do cotidiano, dando origem ao conceito conhecido como “Internet das Coisas” (ATZORI; IERA; MORABITO, 2010), exemplificado pela Figura 5. Sua fundamentação é baseada na interconexão de todos os dispositivos existentes, desde uma simples escova de dente até um sistema de controle de realidade virtual para um apartamento. Com isso, é necessária uma forma de comunicação entre essas ferramentas, ou seja, a necessidade de um protocolo abre margens para um grande fluxo de dados entre esses dispositivos.

Figura 5 – A conexão de dispositivos na rede chamada “Internet das Coisas”.



Fonte: Elaborada pelo autor.

Entretanto, a tecnologia ainda possui limites físicos e continua esbarrando em alguns paradigmas, sendo um deles relacionado à complexidade de resolução de problemas. À medida que a quantidade de dados armazenados aumenta, a complexidade do espaço de busca também acaba por se tornar maior, sendo necessárias análises mais minuciosas bem como um uso de ferramentas mais específicas para se concluir a abordagem do problema.

Esta necessidade de formas alternativas para enfrentar a tarefa de mineração de dados é a principal motivação deste projeto de pesquisa, ao passo que é de grande almejo para o autor entender as atuais técnicas utilizadas para se minerar dados e, com isso, utilizar o conhecimento obtido de outros projetos nas diferentes áreas da computação.

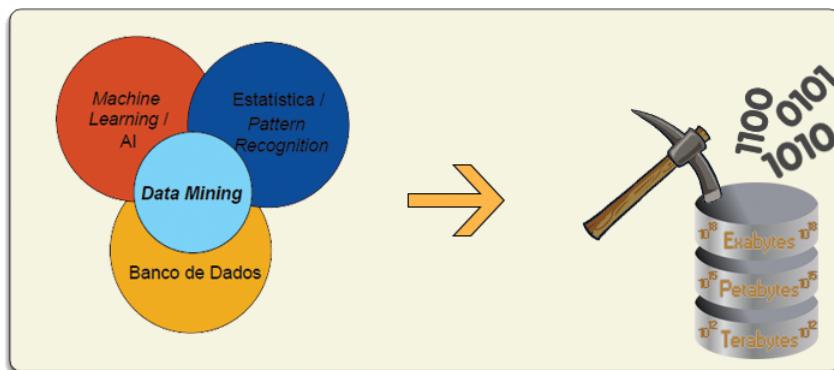
O estudo e a possibilidade do aprimoramento de técnicas de mineração de dados irá proporcionar à comunidade acadêmica e ao autor um maior conhecimento acerca dos mais variados assuntos pesquisados nos dias de hoje. Técnicas evolucionistas, meta-heurísticas, redes neurais, reconhecimento de padrões, e a própria mineração de dados que acaba por envolver essas diversas áreas, estão sofrendo constantes pesquisas, dado que a recuperação e associação de dados armazenados e a possível previsão de futuros acontecimentos são fatores que incentivam o seu desenvolvimento. Do ponto de vista comercial, a mineração de dados pode ser considerada de grande valia para as empresas, trazendo uma grande demanda por profissionais nesta área.

O foco deste trabalho de conclusão de curso dar-se-á em torno da utilização de regras de associação para a mineração de dados, advindas do modelo descritivo. Conforme descritos nos trabalhos de Agrawal ([AGRAWAL; IMIELINSKI; SWAMI, 1993b](#); [AGRAWAL; SRIKANT, 1994](#); [AGRAWAL; PSAILA, 1995](#)) e sucintamente no trabalho de Liu ([LIU; HSU; MA, 1998](#)), a técnica de regra de associação traz consigo indicadores de suporte e confiança para maximizar a eficácia da informação associada. Sendo assim, um fator importante para a aplicação em nosso problema real, dado que queremos trazer uma certa visibilidade ao que realmente é possível se fazer com a mineração de dados.

## 2 Fundamentação Teórica

Nesta seção, é apresentado o embasamento teórico para o desenvolvimento do projeto. Como inicialmente contemplado pela pesquisa, o foco principal deste projeto é a mineração de dados de bases de dados extensas via regras de associação (AGRAWAL; IMIELINSKI; SWAMI, 1993b; TAN; STEINBACH; KUMAR, 2005). Portanto, é apresentada uma introdução sobre o modelo formal que conceitua o que é uma regra de associação, permitindo o entendimento das estruturas básicas aplicadas pelos algoritmos a serem estudados. Após a conceituação da ideia principal do projeto, estenderemos o estudo para a atual esfera da pesquisa, ou seja, também serão descritas formas e respostas para o problema da busca e mineração em espaços de dados muito grandes e complexos. Finalmente, introduziremos os fundamentos necessários para o estudo e operação de alguns dos principais algoritmos (AGRAWAL et al., 1996) utilizados na presente abordagem, para que posteriormente os mesmos possam ser implementados e validados no contexto proposto pelo projeto.

Figura 6 – Ilustração de um simples processo de mineração em uma base de dados.



Fonte: Elaborada pelo autor.

### 2.1 Conceitos em Mineração de Dados

A tecnologia de banco de dados vem sendo extensamente utilizada para o processamento de dados dentro do mundo comercial (PIATESKI; FRAWLEY, 1991), exigindo uma crescente demanda de inovações e domínios de aplicações. Uma área de aplicação que está recebendo bastante atenção nesses últimos anos é a mineração de dados (AGRAWAL; IMIELINSKI; SWAMI, 1993b), processo ilustrado pela Figura 6. Também é possível notar um aumento gradual no número de organizações em busca desses dados comerciais (dados de consumidores, histórico de transações e vendas), armazenando-os e gerando imensas bases de dados, na ordem de terabytes. Tal base contém um grande potencial de mineração em busca de informações valiosas que,

quando postas de forma conexa, poderão gerar conhecimentos inimagináveis ([HOLSHEIMER et al., 1995](#)).

Inicialmente, técnicas de estatística e aprendizado de máquina ainda não possuíam o desejado poder computacional para serem utilizadas nessa busca ([HAN et al., 1996](#)), principalmente quando confrontadas com massivas quantidades de dados. Por isso, antes de estabelecermos um conceito formal sobre a técnica utilizada no projeto para se minerar dados, as chamadas regras de associação, é preciso fazer uma pequena descrição sobre as três principais classes abordadas que envolvem os problemas de mineração de dados e que, consequentemente, também envolvam o problema real e sintético utilizados no projeto.

### 2.1.1 Classificação

O problema de classificação envolve encontrar regras que dividem os dados do problema em grupos disjuntos, por exemplo, o problema de localização de uma loja. É determinado que o sucesso de um empreendimento é influenciado pelas características do bairro em que está localizado, e é de desejo da empresa encontrar prósperos bairros para uma possível localização de uma nova loja. A empresa parte do acesso aos dados do bairro, de suas lojas e comércio atuais e cria um perfil para cada uma dessas lojas baseados em seus dados, categorizando-as e utilizando os perfis para prover uma maior taxa de sucesso ao seu negócio. Outras aplicações podem envolver a utilização de dados de indivíduos, como seguro de saúde e linha de crédito, dentre outros.

Uma variação do problema de classificação é o BestN ([AGRAWAL et al., 1992](#)). Uma empresa pode estar interessada em encontrar os melhores N candidatos que devem receber um produto por correio. Primeiramente, uma pequena quantidade de produtos são enviados para uma seleta parte da população, construindo possíveis perfis positivos de vizinhança. Tais perfis são usualmente construídos a partir de disjunções de conjunções de intervalos de valores de atributos, caracterizando indivíduos na população. Por exemplo, um perfil de correspondência de um produto como um pacote de tacos de golfe pode ser a união de indivíduos com idade entre 30 e 50 anos com renda maior que R\$ 60,000 por ano, e com todos os indivíduos possuindo um carro esportivo ou um relógio de grife. As duas condições geram regras que possuem uma determinada condição, como o antecedente da regra e uma resposta positiva como o consequente. O fator de confiança pode ser associado com cada termo da disjunção e utilizado para desenvolver uma ordem de aplicação dos termos da disjunção, assim obtendo os melhores N candidatos.

### 2.1.2 Associações

Consideremos um problema de supermercado o qual possui uma base de dados de itens comprados por um consumidor uma única vez, conceito tido por **transação** neste exemplo. É

de interesse encontrar associações entre conjuntos de itens com um valor mínimo de **confiança** para esta regra. Um exemplo de associação é que 85% das transações que compram pão e manteiga também compram queijo. O antecedente desta regra consiste no pão e na manteiga e o consequente consiste no queijo sozinho. O valor de 85% é o fator de confiança da regra. Usualmente, é de maior desejo possuir um conjunto de regras com alguma especificação inicial. Outros exemplos que envolvem a problemática do encontro de associações estão descritos abaixo:

- a) Encontrar todas as regras que possuem “Coca-Cola” como consequente. Essas regras irão ajudar uma loja a planejar o que fazer para incrementar as vendas de Coca-Cola;
- b) Encontrar todas as regras que descrevem itens localizados nas prateleiras A e B de uma loja. Essas regras auxiliam no planejamento da disposição de produtos, determinando se a venda de itens na prateleira A está de alguma forma relacionada com a venda de itens na prateleira B; e
- c) Encontrar as  $k$  melhores regras que possuem “presunto” no seu consequente, isto é, encontrar as  $k$  regras que satisfazem os limites mínimos de suporte e confiança do problema e que são úteis para uma aplicação prática. Com isso, podemos entender melhor a definição das variáveis de confiança ou suporte das regras, devidamente descritos na Seção 2.2.

Note que uma transação, necessariamente, não consiste de itens comprados em um mesmo tempo. Ela pode conter informações de períodos acumulados, comprados pelo consumidor ao longo de determinado período.

### 2.1.3 Sequências

Outra grande fonte de problemas de mineração são os dados ordenados, por exemplo, dados temporais relacionados à bolsa de valores e pontos de venda. Um possível exemplo de regra sobre dados do mercado da bolsa de valores é descrito a seguir: quando a ação A está em alta por dois dias consecutivos e a ação B não está em queda nesse período, a ação A tende a continuar em alta no próximo dia por volta de 75% do tempo analisado.

## 2.2 Regras de Associação

O problema da mineração de regras de associação sobre dados comerciais foi introduzido por Agrawal ([AGRAWAL; IMIELINSKI; SWAMI, 1993b](#)). Uma possível situação para tal atividade é a descoberta de uma regra, por exemplo, a qual associa que 63% dos consumidores que compram livros ou revistas em uma livraria também desejam e tomam um café no local.

Tabela 1 – Exemplo de transações de mercado.

<b>TID</b>	<b>Conjunto de Itens</b>
1	{Pão, Leite}
2	{Pão, Fralda, Cerveja, Ovos}
3	{Leite, Fralda, Cerveja, Coca-Cola}
4	{Pão, Leite, Fralda, Cerveja}
5	{Pão, Leite, Fralda, Coca-Cola}

Fonte: Elaborada pelo autor.

Considere os dados apresentados pela Tabela 1, comumente conhecidos por **transações de mercado**. Cada linha dessa tabela corresponde à uma transação, a qual contém um identificador único nomeado por *TID* e um conjunto de itens comprados por um determinado consumidor. A partir desses dados é possível criar um conjunto de regras ou associações para estabelecer vínculos entre os itens desejados. A título de exemplificação, a seguinte regra pode ser extraída do conjunto de dados da Tabela 1:

$$\{Fralda\} \longrightarrow \{Cerveja\}.$$

Essa regra indica um forte relacionamento entre essas duas entidades devido ao fato de que em um mercado muitos consumidores que compram fralda também compram cerveja.

Atualmente, é de grande interesse analisar tais dados e tentar criar relacionamentos a partir deles, construindo informações mais concretas sobre o perfil de seu consumidor, perfil de suas vendas, dentre outras aplicações. Tal garimpo de regras é de extrema importância comercial para a empresa que a utiliza e outro valioso fator nesse processo é o tempo desprendido para se realizar tal tarefa. A necessidade de agilidade no procedimento e o extenso crescimento do tamanho das bases de dados (TOIVONEN, 1996; SAVASERE; OMIECINSKI; NAVATHE, 1995) incentivou o estudo e a pesquisa nessa área, gerando proposições e algoritmos mais eficazes e eficientes (MANNILA; TOIVONEN; VERKAMO, 1994; ZAKI et al., 1997).

A seguir, será introduzido o estudo conceituando o modelo formal sobre o qual as regras de associação se baseiam, bem como será descrita a base matemática utilizada por trás dessa abordagem. Após, o problema será atacado por esse trabalho e descrito gradualmente até os algoritmos utilizados para desenvolver essa técnica. Ao longo da fundamentação algumas imagens e proposições matemáticas foram retiradas e adaptadas do livro “*Introduction to Data Mining*” (TAN; STEINBACH; KUMAR, 2005) para complementar este projeto.

### 2.2.1 Modelo formal

Primeiramente, como a ideia é efetuar uma análise de transações de mercado, é preciso estabelecer um padrão para os dados de entrada. Como apresentado na Tabela 2, cada linha

corresponde à uma transação e cada coluna à um item. Cada item pode ser tratado como uma variável binária com valor 1 caso este item esteja presente na transação e valor 0 caso contrário. Embora essa representação seja bem simplista, não levando em consideração importantes aspectos dos dados como a quantidade de itens vendidos ou o preço pago por sua compra, ela é extremamente funcional para a descrição do problema.

Tabela 2 – Representação binária das transações de mercado.

TID	Pão	Leite	Fralda	Cerveja	Ovos	Coca-Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Fonte: Elaborada pelo autor.

Seja  $I = \{\mathbf{I}_1, \mathbf{I}_2, \dots, \mathbf{I}_N\}$  um conjunto de atributos binários, nomeados de **itens**. Seja  $T$  uma base de dados de transações. Cada transação  $t_i$  é representada por um vetor binário, com  $t_i[k] = 1$  se o item  $I_k$  tiver sido comprado e  $t_i[k] = 0$  caso contrário,  $k = \{1, 2, \dots, M_i\}$ , onde  $M_i$  denota o número de itens da transação  $t_i$ . Existe uma tupla na base de dados para cada transação. Seja  $X$  um conjunto de determinados itens de  $I$ . É dito que uma transação  $t$  satisfaz  $X$  se, para todos os itens  $I_k \in X$ ,  $t[k] = 1$ . Uma importante propriedade desse conjunto de itens é a contagem de seu suporte, a qual se refere ao número de transações que contém um conjunto de itens em particular. Matematicamente, a contagem de suporte,  $\sigma(X)$ , para um conjunto de itens  $X$ , pode ser descrita como segue:

$$\sigma(X) = |\{t_i \mid X \subseteq t_i, t_i \in T\}|, \quad (2.1)$$

onde o símbolo  $|\cdot|$  denota o número de elementos em um conjunto. Para o conjunto de dados demonstrado na Tabela 2, a contagem de suporte para  $\{Cerveja, Fralda, Leite\}$  é igual a dois pois há apenas duas transações que contém esses três itens.

Uma **regra de associação** implica a forma  $X \rightarrow Y$ , onde  $X$  e  $Y$  são conjuntos de itens disjuntos, ou seja,  $X \cap Y = \emptyset$ . A força de uma regra pode ser mensurada em termos de **suporte** e **confiança**. O suporte determina o quão frequente a regra é aplicável ao conjunto de dados utilizado, enquanto a confiança estabelece o quão frequente itens de  $Y$  aparecem em transações que contenham  $X$ . As definições formais e matemáticas dessas métricas são apresentadas abaixo:

$$Suporte; s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}, \quad (2.2)$$

$$Confiança; c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}. \quad (2.3)$$

A regra  $X \rightarrow Y$  é satisfeita para o grupo de transações  $T$  com um fator de confiança  $0 \leq c \leq 1$  caso existam pelo menos  $c\%$  de transações em  $T$  que satisfaçam  $X$  e  $Y$ . A notação  $X \rightarrow Y | c$  será utilizada para especificar se a regra  $X \rightarrow Y$  possui um fator de confiança  $c$ .

**Exemplo:** Considere a regra  $\{Pão, Leite\} \rightarrow \{Fralda\}$ . A contagem de suporte para o conjunto de itens  $\{Pão, Leite, Fralda\}$  é 2 e o número total de transações é 5, portanto o suporte da regra é  $2/5 = 0.4$ . A confiança da regra é obtida através da divisão da contagem de suporte de  $\{Pão, Leite, Fralda\}$  pela contagem de suporte de  $\{Pão, Leite\}$ . Dado que são 3 transações que possuem pão e leite, a confiança para essa regra é  $2/3 = 0.67$ .

Dado um conjunto de transações  $T$ , o interesse principal está contido em gerar todas as regras que satisfaçam certas restrições adicionais ([SRIKANT; VU; AGRAWAL, 1997](#)) referenciadas em duas diferentes formas:

- a) **Restrição Sintática.** Essas restrições envolvem itens que podem aparecer em uma regra. Por exemplo, estamos interessados em regras que contenham apenas um item específico  $I_x$  no consequente ou regras que contenham um específico item  $I_y$  que também já esteja no antecedente. Combinações das restrições descritas também é possível, sendo necessária a requisição, por exemplo, de todas as regras que contenham itens de um conjunto de itens pré-definido  $X$  contidos no consequente e itens de outro conjunto de itens  $Y$  contidos no antecedente;
- b) **Restrição de Suporte.** Essas restrições dizem respeito ao número de transações em  $T$  que suportam uma regra. O suporte de uma regra é definido como a fração de transações em  $T$  que satisfazem a união dos itens contidos no consequente e no antecedente de uma regra de associação. Suporte não deve ser confundido com confiança. Enquanto confiança é a forma de mensurar a “força” de uma regra, o suporte corresponde à significância estatística. Outra motivação para as restrições de suporte advém do fato que é de maior interesse regras que contenham um índice de suporte acima de um determinado limite. Se o suporte não for alto o suficiente significa que a regra não possui valor a ser considerado ou que deve ser escolhida com uma menor preferência (pode ser considerada no futuro).

Com a formulação deste modelo, o problema de mineração de regras pode ser decomposto em dois subproblemas:

- a) Gerar todas as combinações de itens que contenham um nível de suporte transacional acima de um certo limite, denominado *minsupport*. Chamar essas combinações de

conjuntos de itens *frequentes* e todos os outros que não tenham satisfeito o limite de conjuntos de itens *infrequentes*. Restrições sintáticas posteriormente restringem combinações admissíveis. Por exemplo, se apenas regras envolvendo um item  $I_x$  contido no antecedente sejam de interesse, então isso é suficiente para gerar apenas combinações que contenham  $I_x$ ; e

- b) Para um dado conjunto de item *frequente*  $Y = \{I_1, I_2, \dots, I_k\}$ ,  $k \geq 2$ , gerar todas as regras (máximo  $k$  regras) que utilizam os itens do conjunto  $\{I_1, I_2, \dots, I_k\}$ . O antecedente de cada uma dessas regras será um subconjunto  $X$  de  $Y$  tal que  $X$  contenha  $(k - 1)$  itens e o consequente será o item  $Y - X$ . Para gerar uma regra  $X \rightarrow I_j | c$ , onde  $X = \{I_1, I_2, \dots, I_{j-1}, I_{j+1}, \dots, I_k\}$ , tome o suporte de  $T$  e divida pelo suporte de  $X$ . Se a razão for maior que  $c$ , então a regra está satisfeita com fator de confiança  $c$ , caso contrário não estará satisfeita. Também é de grande interesse estabelecer um limite mínimo para a confiança das regras, denominado *minconf*. Note que se um conjunto de itens  $Y$  é frequente, então todo subconjunto de  $Y$  obtido a partir do resultado da solução do primeiro subproblema também será frequente, devendo ter seu suporte disponível e contado. Adicionalmente, todas as regras derivadas de  $Y$  devem satisfazer a restrição de suporte, pois  $Y$  satisfaz a restrição de suporte e  $Y$  é a união dos itens do consequente e antecedente de cada regra dada.

Uma abordagem de força-bruta para minerar regras de associações é computar o suporte e a confiança de toda possível regra. Infelizmente, esse procedimento é altamente custoso devido ao fator exponencial da quantidade de regras extraídas de um conjunto de dados. Particularmente, o número total  $R$  de possíveis regras extraídas de um conjunto de dados que contém  $d$  itens é:

$$R = 3^d - 2^{d+1} + 1. \quad (2.4)$$

Mesmo para pequenos conjuntos de dados, como o da Tabela 1, esse método requer o cálculo de suporte e confiança para  $3^6 - 2^7 + 1 = 602$  regras. Mais de 80% das regras são descartadas após a aplicação de um  $\text{minsupport} = 20\%$  e  $\text{minconf} = 50\%$ , tornando a maioria dos cálculos um desperdício. Para evitar contagens desnecessárias, é de grande valia “podar” as regras logo no início, previamente ao cálculo de seus valores de suporte e confiança.

Um passo inicial para aperfeiçoar a performance dos algoritmos de mineração de regras de associação é dissociar os requerimentos de suporte e confiança. A partir da Equação 2.3, note que o suporte da regra  $X \rightarrow Y$  depende apenas do suporte de seu conjunto correspondente de itens,  $X \cup Y$ . Por exemplo, as regras a seguir possuem suporte idêntico devido ao fato de envolverem itens do mesmo conjunto de itens,  $\{\text{Cerveja}, \text{Fralda}, \text{Leite}\}$ :

$$\begin{aligned} \{Cerveja, Fralda\} &\rightarrow \{Leite\}, \{Cerveja, Leite\} \rightarrow \{Fralda\}, \\ \{Fralda, Leite\} &\rightarrow \{Cerveja\}, \{Cerveja\} \rightarrow \{Fralda, Leite\}, \\ \{Leite\} &\rightarrow \{Cerveja, Fralda\}, \{Fralda\} \rightarrow \{Cerveja, Leite\}. \end{aligned}$$

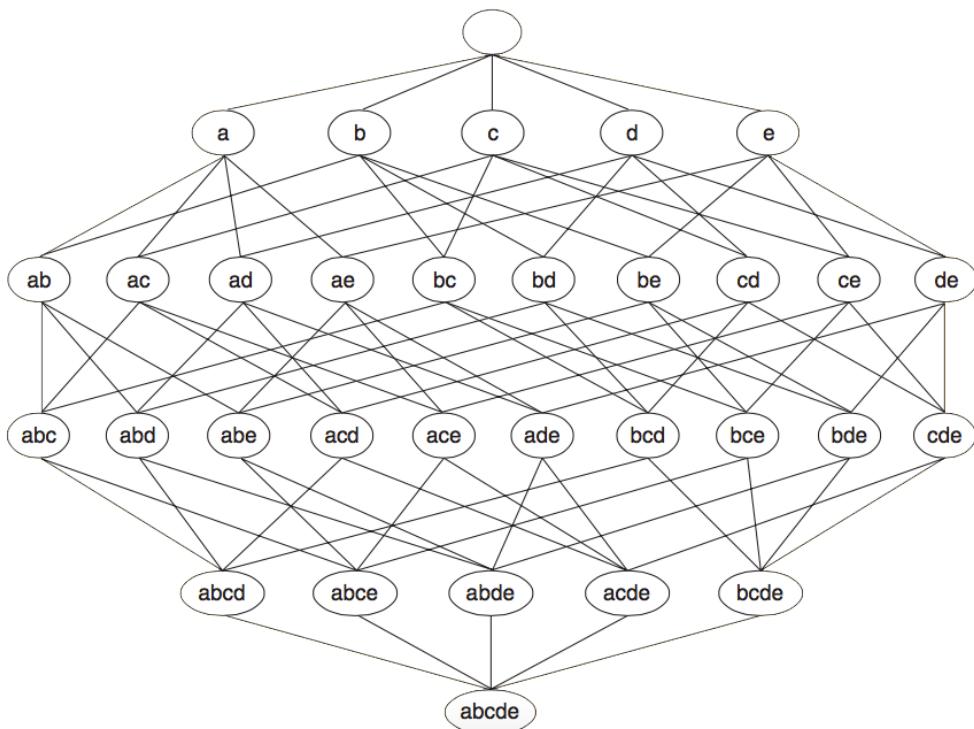
Se um conjunto de itens é infrequente, então todas as regras candidatas podem ser cortadas imediatamente sem a necessidade de computar os seus valores de confiança.

Portanto, havendo determinado os conjuntos de itens frequentes, a solução para o segundo subproblema é intuitiva e direta. Na próxima seção serão descritas técnicas eficientes focadas na primeira parte da resolução do problema, ou seja, a geração de conjuntos de itens frequentes. Enquanto isso, serão também apresentados diferentes algoritmos e seus procedimentos para a geração desses conjuntos.

### 2.2.2 Geração de conjuntos de itens frequentes

A Figura 7, composta por uma estrutura em grade, pode ser utilizada para enumerar uma lista de todos os possíveis conjuntos de itens. Neste exemplo, o conjunto de itens da grade é representado por  $I = \{a, b, c, d, e\}$ . Usualmente, uma base de dados que contenha  $k$  itens pode gerar até no máximo  $2^k - 1$  conjuntos de itens frequentes, excluindo o conjunto nulo. Dado que  $k$  pode ser extremamente grande nas mais variadas aplicações reais, o espaço de busca dos conjuntos de itens que necessita ser explorado também é exponencialmente grande.

Figura 7 – Estrutura em grade de um conjunto de itens.

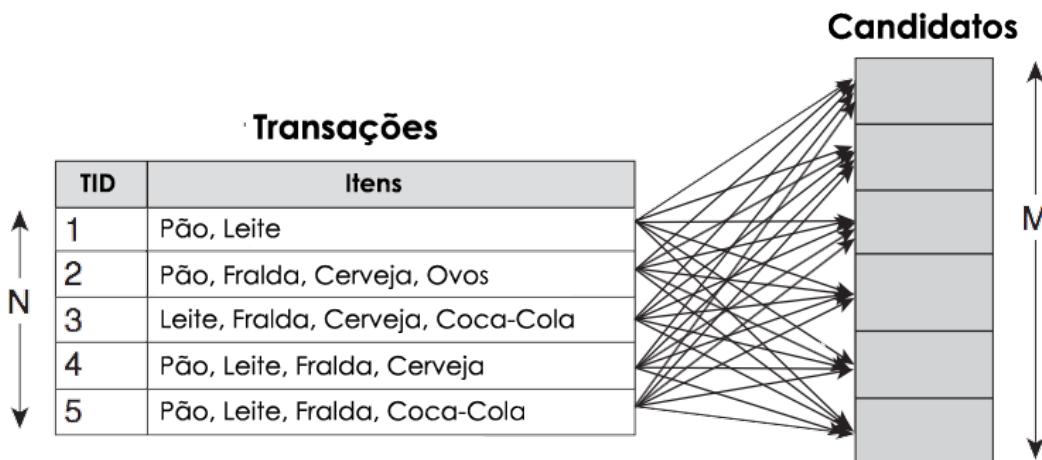


Fonte: ([TAN; STEINBACH; KUMAR, 2005](#)).

Os algoritmos para a busca de conjuntos de itens frequentes efetuam múltiplas iterações sobre a base de dados, onde na primeira iteração é calculado o suporte individual dos itens e estabelecidos quais deles são “frequentes”, i.e. possuem suporte mínimo. Em cada passo subsequente é utilizado um conjunto de itens frequente obtido no passo anterior. A partir desse conjunto, novos conjuntos de itens potencialmente frequentes chamados de conjuntos de itens *candidatos* são gerados, e então é calculado o atual suporte desses conjuntos candidatos. No final deste procedimento, são determinados quais conjuntos candidatos realmente são conjuntos frequentes e que podem ser considerados para a próxima iteração. Este processo repete-se até que não existam mais novos conjuntos de itens frequentes.

Uma metodologia de força-bruta é utilizada para determinar a contagem de suporte de cada conjunto de itens candidatos na estrutura em grade. Para isso, é necessário comparar todo candidato contra cada transação, operação demonstrada pela Figura 8. Se o candidato está contido na transação, a sua contagem de suporte será incrementada. Por exemplo, o suporte para  $\{Leite, Pão\}$  é incrementado três vezes por conta do conjunto de itens estar contido nas transações 1, 4 e 5. Tal abordagem é extremamente custosa porque requer  $O(MNw)$  comparações, onde  $N$  é o número de transações,  $M = 2^k - 1$  é o número de conjuntos de itens candidatos e  $w$  é o tamanho máximo da transação.

Figura 8 – Contagem do suporte de conjunto de itens candidatos.



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

Existem diversas maneiras para se reduzir a complexidade computacional da geração de conjuntos de itens frequentes, dentre elas:

- Redução do número de conjuntos de itens candidatos ( $M$ )**. O princípio *Apriori*, descrito na próxima seção, é um método efetivo para eliminar alguns dos conjuntos de itens candidatos sem a contagem de seus valores de suporte; e
- Redução do número de comparações**. Ao invés de corresponder todo conjunto de itens candidatos contra toda transação, é possível reduzir o número de com-

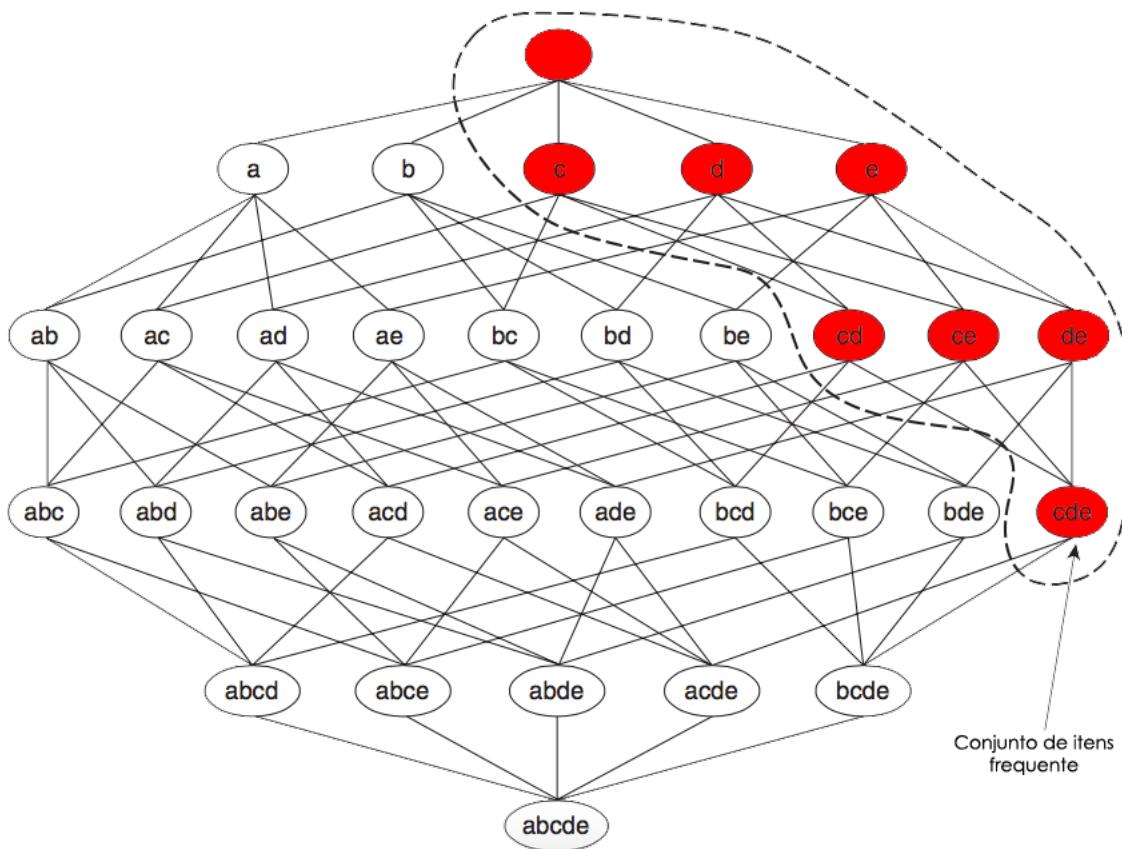
parações utilizando estruturas de dados mais avançadas, tanto para armazenar os conjuntos de itens candidatos como também para comprimir a base de dados. Esses métodos serão discutidos posteriormente.

### Princípio *Apriori*

**Teorema:** *Se um conjunto de itens é frequente, então todos os seus subconjuntos também devem ser frequentes.*

Para ilustrar a ideia acima, considere a grade do conjunto de itens exposta na Figura 9. Suponha que  $\{c, d, e\}$  seja um conjunto de itens frequente. Toda transação que contenha  $\{c, d, e\}$  também deve conter seus subconjuntos  $\{c, d\}$ ,  $\{c, e\}$ ,  $\{d, e\}$ ,  $\{c\}$ ,  $\{d\}$  e  $\{e\}$ . Consequentemente, se  $\{c, d, e\}$  é frequente, então todos os subconjuntos de  $\{c, d, e\}$  também são frequentes.

Figura 9 – Ilustração do princípio *Apriori*. Caso  $\{c, d, e\}$  seja frequente, todos seus subconjuntos serão frequentes.

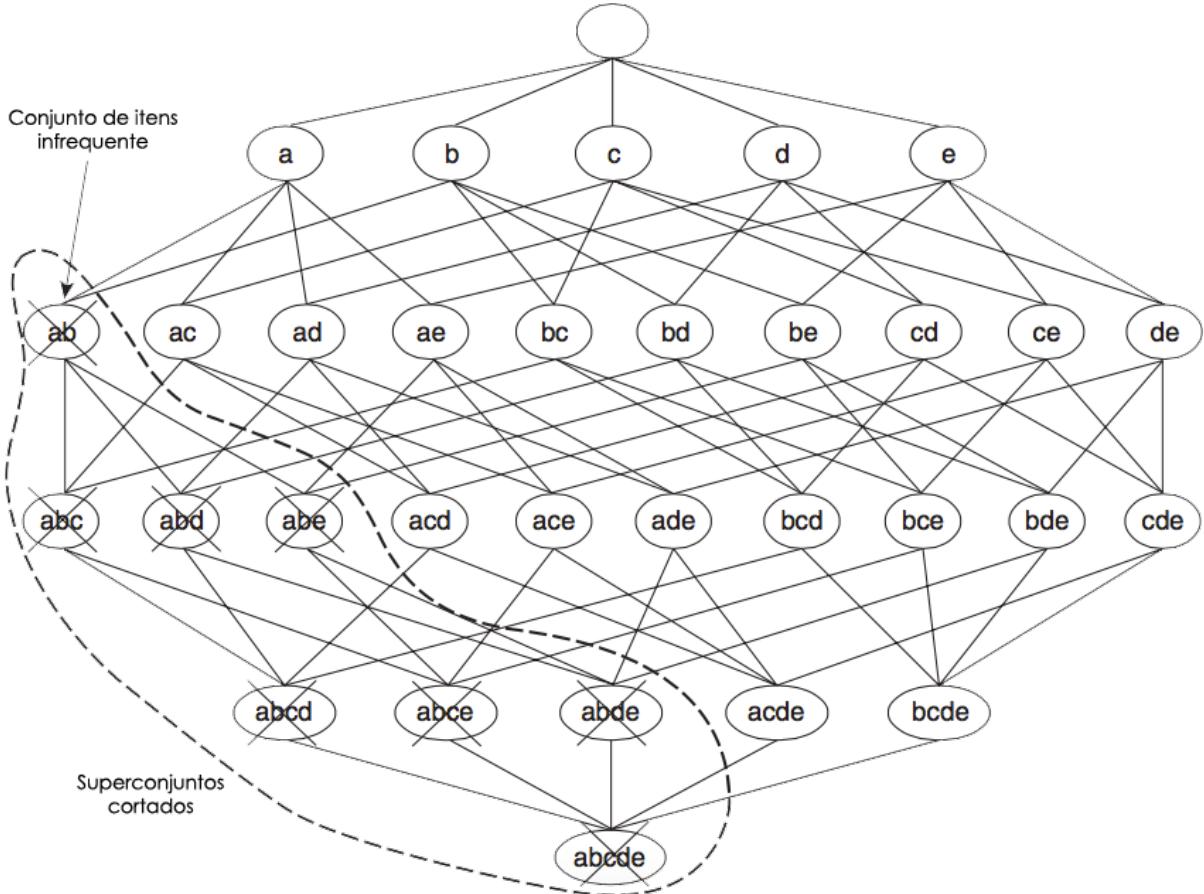


Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

Por outro lado, se um conjunto de itens como  $\{a, b\}$  for infrequente, então todos os seus superconjuntos também devem ser infrequentes. Como ilustrado na Figura 10, todo o subgrafo contendo os superconjuntos de  $\{a, b\}$  podem ser cortados imediatamente após a descoberta de que o conjunto  $\{a, b\}$  é infrequente. Essa estratégia de corte no espaço de busca

exponencial baseado na métrica de suporte é conhecida por **corte baseado no suporte**. Tal método é apenas possível por conta de uma propriedade da medida de suporte, a qual consiste em que o suporte de um conjunto de itens nunca excede o suporte de seus subconjuntos. Essa propriedade também é conhecida por propriedade **anti-monótona** da medida de suporte.

Figura 10 – Ilustração do corte baseado no suporte. Caso  $\{a, b\}$  seja infrequente, todos seus superconjuntos serão infreqüentes.



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

**Definição (Propriedade de Monotonidade):** Seja  $I$  um conjunto de itens, e  $J = 2^I$  uma potência de  $I$  com base 2. Uma medida  $f$  é monótona ou fechada para cima se:

$$\forall X, Y \in J : (X \subseteq Y) \longrightarrow f(X) \leq f(Y), \quad (2.5)$$

a qual significa que se  $X$  for um subconjunto de  $Y$ , então  $f(X)$  não deve exceder  $f(Y)$ . Por outro lado,  $f$  é anti-monótona ou fechada para baixo se:

$$\forall X, Y \in J : (X \subseteq Y) \longrightarrow f(Y) \leq f(X), \quad (2.6)$$

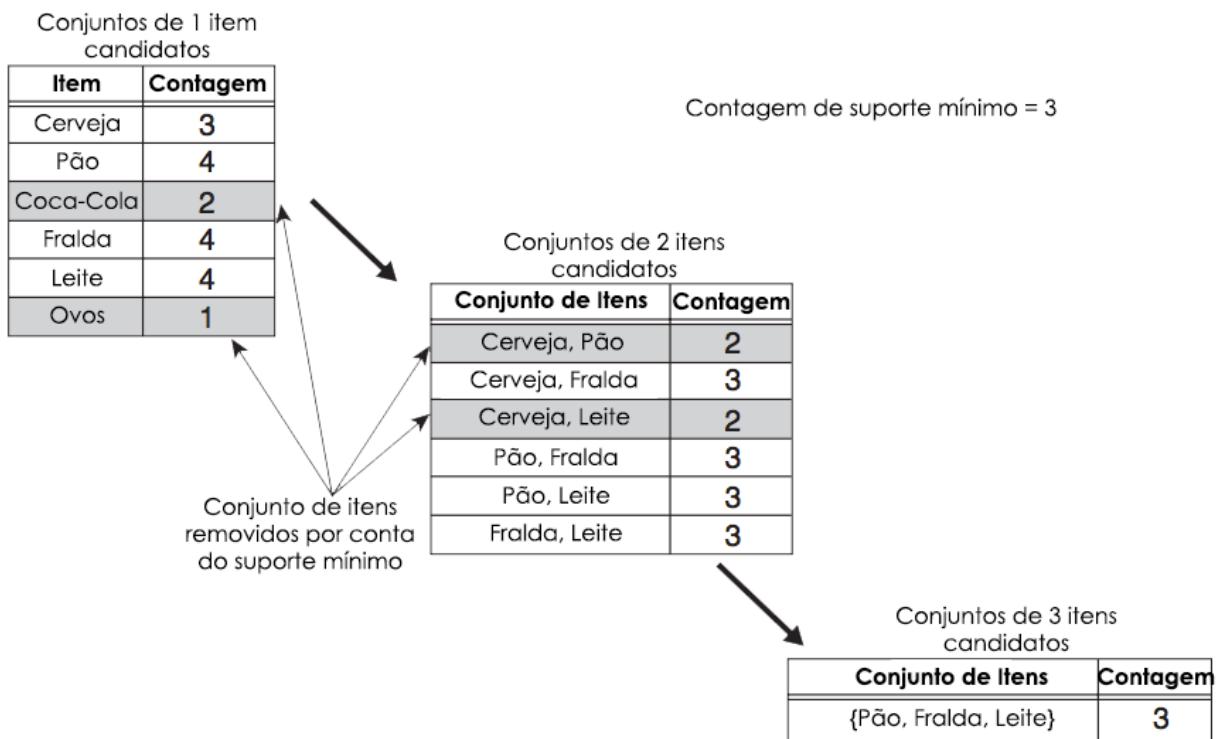
que delimita que se  $X$  for um subconjunto de  $Y$ , então  $f(Y)$  não deve exceder  $f(X)$ .

Qualquer medida que possua uma propriedade anti-monótona pode ser incorporada diretamente ao algoritmo de mineração para cortar efetivamente o espaço de busca exponencial dos conjuntos de itens candidatos.

### Geração de conjuntos de itens frequentes no algoritmo *Apriori*

O primeiro algoritmo de mineração de regras de associação foi o *Apriori*, tido como o precursor do uso do corte baseado no suporte para controlar o crescimento exponencial da geração de conjuntos de itens candidatos. A Figura 11 ilustra o procedimento de geração de conjuntos de itens candidatos frequentes do algoritmo *Apriori* para as transações descritas na Tabela 1. Assumimos que o limite máximo de suporte seja de 60%, o qual é equivalente à contagem mínima de suporte igual a 3.

Figura 11 – Ilustração da geração de conjuntos de itens frequentes pelo algoritmo *Apriori*.



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

Inicialmente, todo item é considerado como um conjunto de um item candidato. Após a contagem de seus suportes, os itens que não chegarem ao suporte mínimo utilizado (Coca-Cola e Ovos) são descartados. Na próxima iteração são gerados conjuntos de dois itens candidatos usando apenas os conjuntos frequentes de um item, essencialmente por conta do teorema proposto. Por conta de existirem apenas quatro conjuntos frequentes de um item, o número de conjuntos candidatos de dois itens gerados pelo algoritmo é  $\binom{4}{2} = 6$ . Dois desses seis candidatos,  $\{Pão, Cerveja\}$  e  $\{Leite, Cerveja\}$ , são tidos como infrequentes. Os quatro

restantes são frequentes e utilizados na geração de conjuntos de três itens. Sem o corte baseado no suporte, tem-se  $\binom{6}{3} = 20$  conjuntos de três itens candidatos gerados. Com o princípio *Apriori*, só é necessário manter os conjuntos de três itens candidatos os quais possuem subconjuntos frequentes. O único candidato que possui essa propriedade é  $\{Pão, Fralda, Leite\}$ .

A eficiência da estratégia de corte do algoritmo *Apriori* pode ser demonstrada a partir da contagem do número de conjuntos de itens candidatos gerados. Uma estratégia de força-bruta para este mesmo exemplo produzirá

$$\binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 6 + 15 + 20 = 41$$

candidatos. Enquanto que com o princípio *Apriori*, esse número cairá para

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13$$

candidatos, representando uma redução de 68% no número de conjuntos de itens candidatos mesmo neste simples exemplo.

O pseudo-código do algoritmo *Apriori* para a geração de conjunto de itens frequentes é apresentado pelo Algoritmo 1. Seja  $C_k$  o conjunto de  $k$  itens candidatos e  $F_k$  o conjunto de  $k$  itens frequentes:

- a) Inicialmente, o algoritmo faz uma simples busca sobre a base de dados para determinar a contagem de suporte de cada item. Após essa etapa, todos os conjuntos frequentes de 1 item,  $F_1$ , serão encontrados;
- b) Logo, o algoritmo gerará iterativamente novos conjuntos de  $k$  itens candidatos usando os conjuntos frequentes de  $(k - 1)$  itens encontrados na iteração anterior. A geração de candidatos está implementada na função *apriorigen*;
- c) Para contar o suporte dos candidatos, o algoritmo precisa percorrer novamente a base de dados. A função de subconjunto é usada para determinar todos os conjuntos de itens candidatos em  $C_k$  que estão contidos em cada transação  $t$ ;
- d) Após a contagem de seus suportes, o algoritmo elimina todos os conjuntos de itens candidatos os quais possuem contagem de suporte inferior ao *minsupport*; e
- e) O algoritmo termina quando não há mais conjuntos de itens frequentes gerados, ou seja,  $F_k = \emptyset$ .

**Algoritmo 1** – GERAÇÃO DO CONJUNTO DE ITENS FREQUENTE DO ALGORITMO *Apriori*

1.  $k = 1$ .
2.  $F_k = \{i \mid i \in I \wedge \sigma(\{i\}) \geq N \times \text{minsupport}\}$ .
3. **Repete**
4.      $k = k + 1$ .
5.      $C_k = \text{apriorigen}(F_{k-1})$ .
6.     **Para** cada transação  $t \in T$  **faz**
7.          $C_t = \text{subconjunto}(C_k, t)$ .
8.         **Para** cada conjunto de item candidato  $c \in C_t$  **faz**
9.              $\sigma(c) = \sigma(c) + 1$ .
- 10.
11.      $F_k = \{c \mid c \in C_k \wedge \sigma(c) \geq N \times \text{minsupport}\}$ .
12.     **até**  $F_k = \emptyset$
13.  $\text{Resposta} = \bigcup F_k$ .

Essa geração de conjuntos de itens frequentes do algoritmo *Apriori* possui duas importantes características. Primeiramente, o algoritmo percorre apenas uma altura por vez, ou seja, ele parte de conjuntos de um item frequente até conjuntos de tamanho  $k_{max}$ , onde  $k_{max}$  é o tamanho máximo dos conjuntos de itens frequentes. Em segundo lugar, ele emprega uma estratégia de geração e teste, onde a cada iteração novos conjuntos de itens candidatos são gerados a partir de conjuntos de itens frequentes encontrados na iteração anterior. É então calculado o suporte para cada candidato e testado para o limite mínimo de suporte (*minsupport*). O número total de iterações necessárias para a convergência do algoritmo é da ordem ( $k_{max} + 1$ ).

### Corte e geração de candidatos

A função *apriorigen* demonstrada no Algoritmo 1 gera novos conjuntos de itens candidatos a partir das seguintes operações:

- Geração de candidatos.** Essa operação gera novos conjuntos de  $k$  itens candidatos baseado nos conjuntos de  $(k - 1)$  itens frequentes encontrados na iteração anterior; e
- Corte de candidatos.** Essa operação elimina alguns dos conjuntos de  $k$  itens candidatos utilizando a metodologia do corte baseado no suporte ([WANG; HE; HAN, 2000](#)).

Para melhor exemplificar a operação de corte, considere um conjunto de  $k$  itens candidato  $X = \{I_1, I_2, \dots, I_k\}$ . O algoritmo deve determinar se todos os seus subconjuntos,

$X \setminus \{I_j\}$ ,  $\forall j = \{1, 2, \dots, k\}$ , são frequentes. Caso um deles seja infrequente, então  $X$  é automaticamente cortado. Essa abordagem pode reduzir com certa eficiência o número de conjuntos de itens candidatos considerados durante a contagem de suporte. A complexidade dessa operação é da ordem de  $O(k)$  para cada conjunto de  $k$  itens candidato. Entretanto, não é preciso examinar todos os  $k$  subconjuntos de um dado conjunto de itens candidato. Se  $m$  dos  $k$  subconjuntos foram utilizados para gerar um candidato, só é necessário checar os restantes  $k - m$  subconjuntos durante a etapa de corte dos candidatos.

Geralmente, existem diversas maneiras de gerar conjuntos de itens candidatos. A lista a seguir apresenta os requerimentos básicos para um eficiente procedimento de geração de candidatos:

- a) Deve-se evitar a geração de muitos candidatos desnecessários. Um conjunto de itens candidato é desnecessário se pelo menos um de seus subconjuntos for infrequente. Tal candidato é tido como infrequente de acordo com a propriedade de monotonicidade do suporte;
- b) Deve-se certificar que o conjunto de candidatos esteja completo, isto é, não foram deixados de lado quaisquer conjuntos de itens frequentes. Para assegurar essa operação, o conjunto dos conjuntos de itens candidatos deve englobar o conjunto de todos os conjuntos de itens frequentes, ou seja,  $\forall k : F_k \subseteq C_k$ ; e
- c) Não se deve gerar o mesmo conjunto de itens candidato mais de uma vez. Por exemplo, o conjunto de itens candidato  $\{a, b, c, d\}$  pode ser obtido através de várias formas, agrupando  $\{b, c\}$  com  $\{a, d\}$ ,  $\{a\}$  com  $\{b, c, d\}$ , dentre outros. A geração duplicada de candidatos proporciona um desperdício e um atraso nos cálculos, devendo ser evitada por razões óbvias.

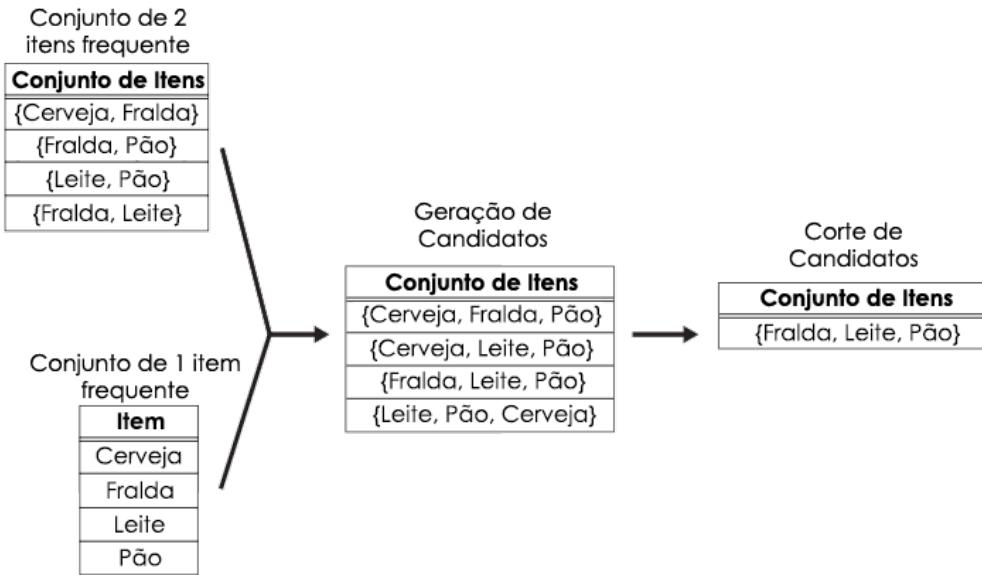
A seguir, são descritos brevemente os diferentes procedimentos de geração de candidatos, incluindo o utilizado pela função *apriorigen*.

**Método  $F_{k-1} \times F_1$ :** Esse método extende cada conjunto de  $(k - 1)$  itens frequente com outros itens frequentes. A Figura 12 ilustra como um conjunto de dois itens frequente,  $\{Fralda, Cerveja\}$ , pode ser aumentado com um item frequente como  $\{Pão\}$  para produzir um conjunto de três itens candidato  $\{Fralda, Cerveja, Pão\}$ . Esse procedimento produzirá  $O(|F_{k-1}| \times |F_1|)$  conjuntos de  $k$  itens candidatos, onde  $|F_j|$  é o número dos conjuntos de  $j$  itens frequentes. A complexidade geral dessa etapa é  $O(\sum_k k |F_{k-1}| |F_1|)$ .

O método é completo porque cada conjunto de  $k$  itens frequente é composto por um conjunto de  $(k - 1)$  itens frequente e um conjunto de um item frequente. Portanto, todos os conjuntos de  $k$  itens frequentes fazem parte dos conjuntos de  $k$  itens candidatos gerados por esse procedimento.

Contudo, essa abordagem não previne o fato do mesmo conjunto de itens candidato ser gerado mais de uma vez. Por exemplo,  $\{Pão, Fralda, Leite\}$  pode ser gerado a partir

Figura 12 – Geração e corte de conjuntos de  $k$  itens candidatos através do método  $F_{k-1} \times F_1$ .



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

da combinação de  $\{Pão, Fralda\}$  com  $\{Leite\}$ ,  $\{Fralda, Leite\}$  com  $\{Pão\}$  ou até mesmo  $\{Pão, Leite\}$  com  $\{Fralda\}$ . Uma maneira de evitar a geração de candidatos duplicados é assegurar que todos os itens de cada conjunto de itens frequente seja mantido na sua ordem lexicográfica. Cada conjunto  $X$  de  $(k - 1)$  itens frequentes é então extendido com os itens frequentes que são lexicograficamente maiores que os itens contidos em  $X$ . Por exemplo, o conjunto de itens  $\{Fralda, Leite\}$  pode ser aumentado com  $\{Pão\}$ , dado que  $\{Pão\}$  é lexicograficamente maior que  $\{Leite\}$  e  $\{Fralda\}$ . Entretanto, não se deve aumentar o conjunto  $\{Leite, Pão\}$  com  $\{Fralda\}$  nem  $\{Fralda, Pão\}$  com  $\{Leite\}$ , pois a condição de ordem lexicográfica seria violada.

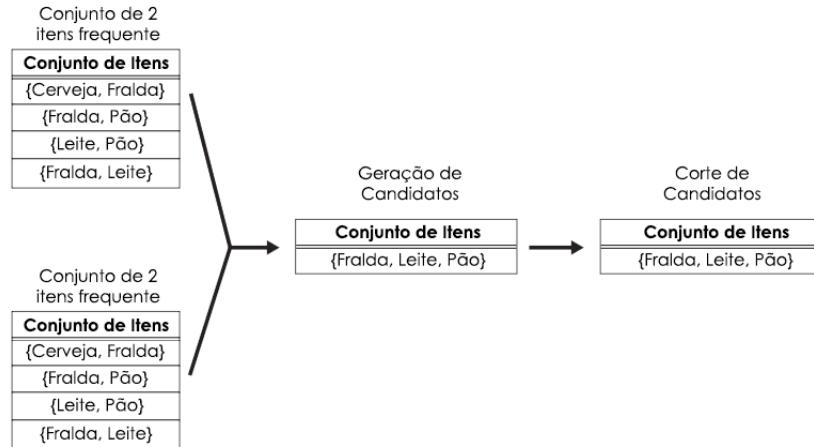
Esse procedimento, mesmo sendo bastante efetivo, ainda é capaz de produzir um número desnecessário de candidatos. Por exemplo, o conjunto de itens candidato obtido através da junção de  $\{Cerveja, Fralda\}$  com  $\{Leite\}$  é desnecessário, porque um de seus subconjuntos,  $\{Cerveja, Leite\}$ , é infrequente.

**Método  $F_{k-1} \times F_{k-1}$ :** O procedimento de geração de candidatos na função *apriorigen* associa um par de conjuntos de  $(k - 1)$  itens frequentes apenas se seus primeiros  $(k - 2)$  itens são iguais. Seja  $A = \{a_1, a_2, \dots, a_{k-1}\}$  e  $B = \{b_1, b_2, \dots, b_{k-1}\}$  um par de conjuntos de  $(k - 1)$  itens frequentes.  $A$  e  $B$  são combinados caso satisfaçam a seguinte condição:

$$a_i = b_i, \forall i = \{1, 2, \dots, k - 2\} \text{ e } a_{k-1} \neq b_{k-1}.$$

Na Figura 13, os conjuntos de itens frequentes  $\{Fralda, Pão\}$  e  $\{Leite, Pão\}$  são combinados para formar um conjunto de três itens candidato  $\{Fralda, Leite, Pão\}$ . O algoritmo

Figura 13 – Geração e corte de conjuntos de  $k$  itens candidatos através do método  $F_{k-1} \times F_{k-1}$ .

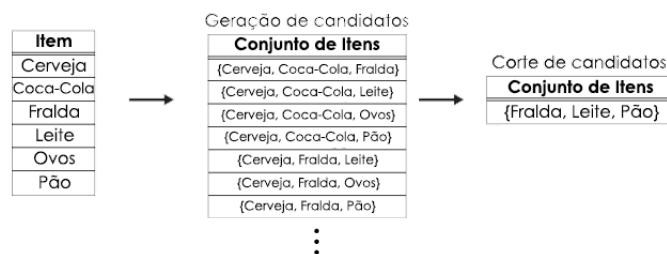


Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

não precisa associar  $\{Cerveja, Fralda\}$  com  $\{Fralda, Leite\}$  porque o primeiro item de cada conjunto de itens é diferente. Caso  $\{Cerveja, Fralda, Leite\}$  fosse um candidato viável, ele seria obtido através da junção de  $\{Cerveja, Fralda\}$  com  $\{Cerveja, Leite\}$ . Entretanto, apesar desse procedimento prevenir a geração de candidatos duplicados através do ordenamento lexicográfico, é necessária uma etapa adicional de corte de candidatos para assegurar com que os restantes  $(k - 2)$  subconjuntos do candidato sejam frequentes, dado que cada candidato é gerado a partir da combinação de conjuntos de  $(k - 1)$  itens frequentes.

**Método da Força-Bruta:** O método da força-bruta considera todo conjunto de  $k$  itens como um candidato potencial e então aplica a etapa de corte de candidatos para remover quaisquer candidatos desnecessários, processo ilustrado pela Figura 14. O número de conjuntos de itens candidatos gerados no nível  $k$  é igual a  $\binom{d}{k}$ , onde  $d$  é o número total de itens. Embora a geração de candidatos seja um procedimento trivial, o corte de candidatos torna-se extremamente custoso por conta do grande número de conjuntos de itens que devem ser examinados. A quantidade de cálculos necessários para cada candidato é  $O(k)$ , e a complexidade total deste método pode ser dada por  $O(\sum_{k=1}^d k \times \binom{d}{k}) = O(d \cdot 2^{d-1})$ .

Figura 14 – Geração e corte de conjuntos de  $k$  itens candidatos através do método de força bruta.



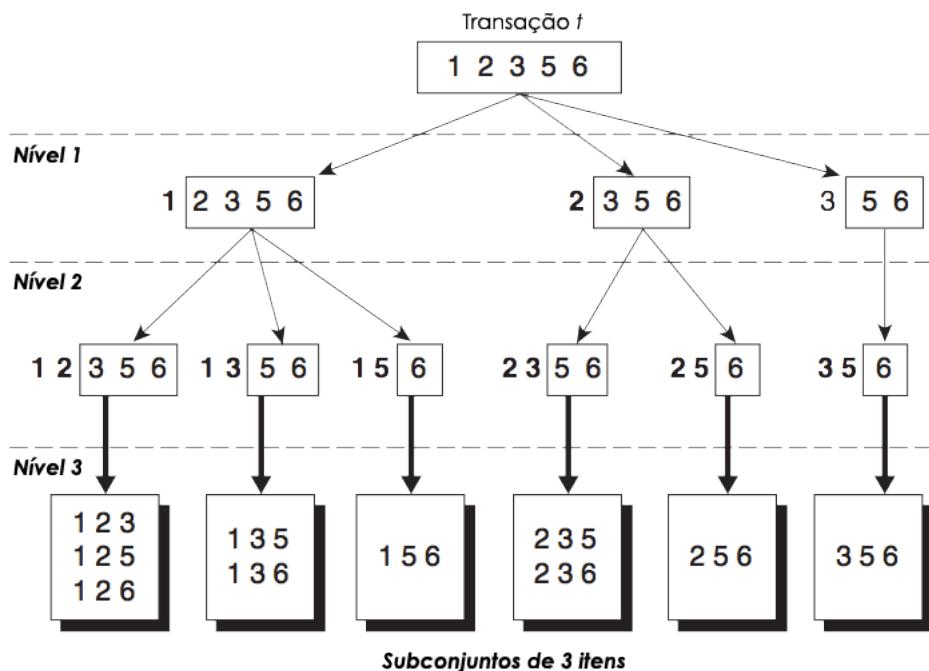
Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

## Contagem de suporte

A contagem de suporte é o processo para determinar a frequência de ocorrência de cada conjunto de itens candidato que tenha sobrevivido à etapa de corte realizada previamente. Uma metodologia proposta para mensurar essa métrica é comparar cada transação com cada conjunto de itens candidato e atualizar as contagens de suporte dos candidatos contidos nessas transações. Essencialmente, essa abordagem é extremamente custosa, sendo ineficiente em casos onde o número de transações e o conjunto de itens candidatos sejam grandes.

Uma alternativa para essa proposição é enumerar todos os conjuntos de itens contidos em cada transação e utilizá-los para atualizar a contagem de suporte de seus respectivos conjuntos de itens candidatos. Considere uma transação  $t$  que possua 5 itens,  $\{1, 2, 3, 5, 6\}$ . Existem  $\binom{5}{3} = 10$  conjuntos de itens de tamanho três contidos nessa transação. Alguns desses conjuntos de itens podem corresponder aos conjuntos de três itens candidatos sob avaliação, os quais possuirão sua contagem de suporte incrementada. Outros subconjuntos de  $t$  que não corresponderem aos conjuntos candidatos poderão ser eliminados.

Figura 15 – Enumeração de subconjuntos de três itens de uma transação  $t$ .



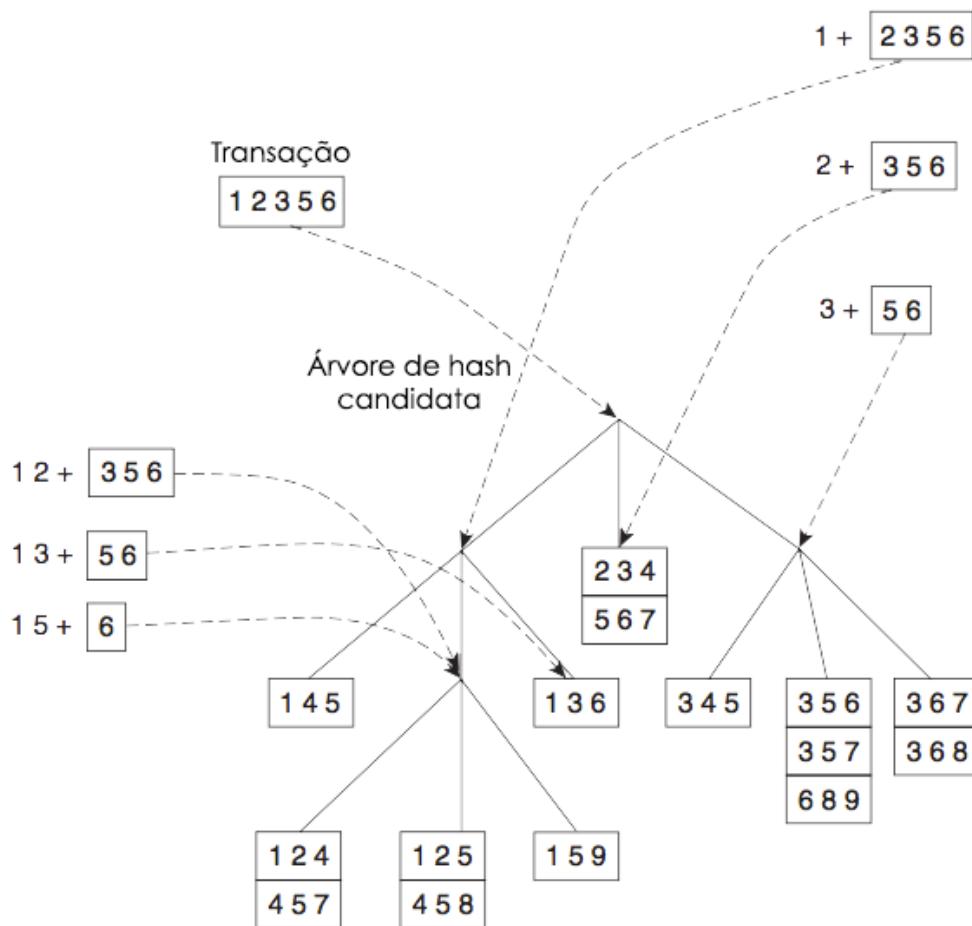
Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

Pode-se observar na Figura 15 como os conjuntos de itens contidos em uma transação podem ser enumerados, especificando seus itens um por um, do item contido mais à esquerda até o mais à direita. Contudo, ainda é necessário determinar se os conjuntos de três itens enumerados correspondem à um conjunto de item candidato. Caso essa afirmação seja verdadeira, a contagem de suporte do candidato correspondente será incrementada. Existem diversas maneiras dessa

estrutura ser implementada mas a seguir veremos uma forma muito eficiente utilizando as estruturas de árvores de *hash* (COFFMAN JR.; EVE, 1970).

Ao examinarmos o algoritmo *Apriori*, os conjuntos de itens candidatos são particionados em diferentes blocos e guardados em um árvore de *hash* (PARK; CHEN; YU, 1995). Durante sua contagem de suporte, conjuntos de itens contidos em cada transação também passarão por uma função *hash* para serem colocados em seus blocos apropriados. Com isso, ao invés de compararmos cada conjunto de itens na transação com cada conjunto de itens candidato, apenas os conjuntos candidatos pertencentes ao mesmo bloco são correspondidos.

Figura 16 – Ilustração da estrutura de uma árvore de *hash*.



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

A Figura 16 mostra um exemplo da estrutura de árvore de *hash*. Cada nó interno da árvore utiliza a seguinte função de *hashing*:

$$h(p) = p \mod 3, \quad (2.7)$$

para determinar qual o próximo ramo do nó atual que deve ser seguido. Por exemplo, os itens 1, 4 e 7 sofrem um *hash* para o mesmo ramo (o mais à esquerda), pois possuem o mesmo resto

após a ativação da função de *hashing*. Todos os conjuntos de itens candidatos são guardados em nós folha da árvore de *hash*.

Considere uma transação  $t = \{1, 2, 3, 5, 6\}$ . Para atualizar a contagem de suporte dos conjuntos de itens candidatos, a árvore de *hash* deve ser percorrida de tal modo que todos os nós folhas contendo conjuntos de três itens candidatos e que pertençam à  $t$  devem ser visitados ao menos uma única vez. É possível de lembrar que os conjuntos de três itens contidos em  $t$  devem começar com os itens 1, 2 ou 3, como indicado pelos prefixos das estruturas de nível 1 demonstrados na Figura 16. Portanto, no nó raiz da árvore de *hash* os itens 1, 2 e 3 de cada transação sofrem a função de *hashing* separadamente. O item 1 sofre um *hash* para o filho à esquerda, o item 2 para o filho central e o item 3 para o filho à direita. No próximo nível da árvore, a transação sofre a função de *hashing* no segundo item indicado pela estrutura de nível 2 da árvore. Por exemplo, após usar o *hash* no item 1 no nó raiz, os itens 2, 3 e 5 da transação irão sofrer o *hash*, onde os itens 2 e 5 serão levados para o filho central e o item 3 novamente sofre um *hash* para o filho à direita. Esse processo continua até que todos os nós folhas da árvore de *hash* tenham sido alcançados. Os conjuntos de itens candidatos guardados nos nós folhas visitados são comparados com a transação. Se um candidato for um subconjunto de transação, então a sua contagem de suporte será incrementada.

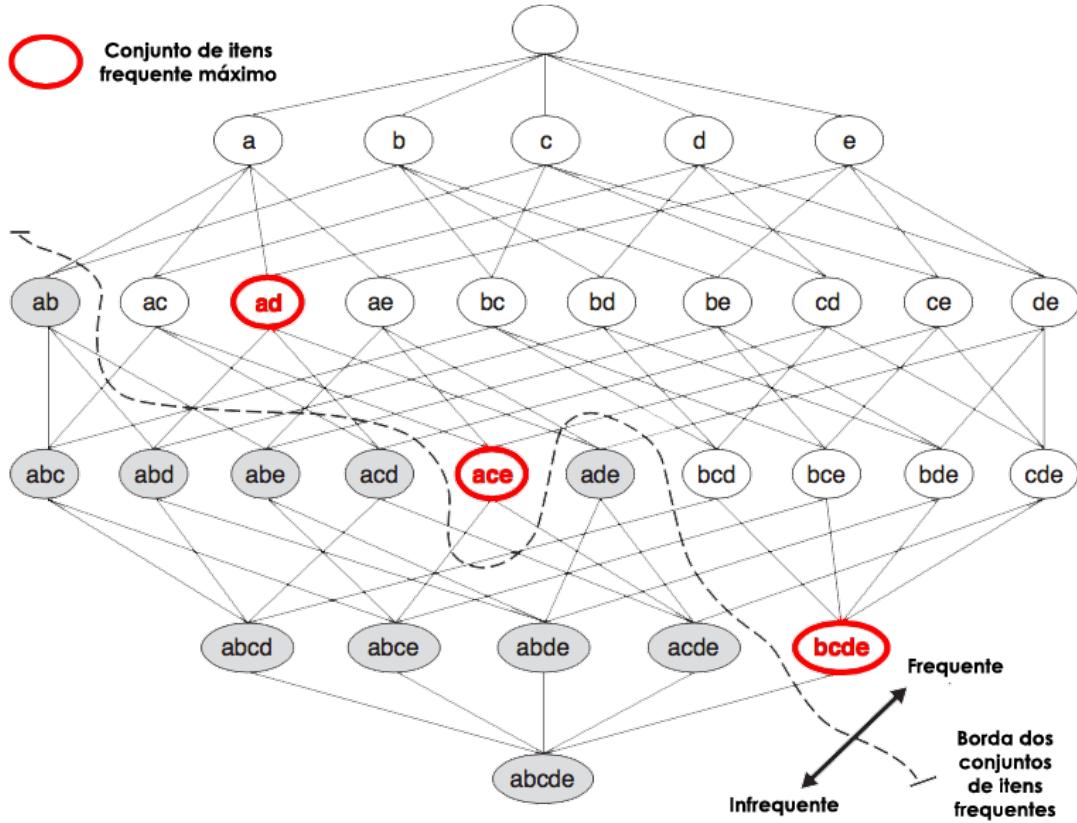
### 2.2.3 Representações de conjuntos de itens frequentes

Atualmente, o tamanho do conjunto de dados transacionais pode ser bastante extenso, produzindo um grande número de conjuntos de itens frequentes. Devido a esse fato, é de grande interesse identificar pequenas representações de conjuntos de itens (PASQUIER et al., 1999), os quais a partir de sua derivação são capazes de gerar todos os outros conjuntos de itens frequentes. Outro importante conceito para o problema é a definição de *superconjuntos*, por exemplo, um conjunto  $X$  é superconjunto de  $Y$  se  $Y \subset X$ .

#### **Conjuntos de itens frequentes máximo**

Um conjunto de itens frequentes máximo é definido como um conjunto de itens frequente o qual não possui superconjuntos imediatos frequentes. Considere a grade de itens demonstrada na Figura 17. Os conjuntos de itens pertencentes à grade são divididos em dois grupos: frequentes e infrequentes. Uma borda para a divisão desses grupos também é representada na figura por uma linha pontilhada. Cada conjunto de itens localizado acima da borda é considerado frequente enquanto aqueles localizados abaixo dela são considerados infrequentes. Dentre os conjuntos de itens que situam-se ao longo da borda,  $\{a, d\}$ ,  $\{a, c, e\}$  e  $\{b, c, d, e\}$ , são considerados conjuntos de itens frequentes máximos porque os seus superconjuntos imediatos são infrequentes. Um conjunto de itens como  $\{a, d\}$  é frequente e máximo porque todos os seus superconjuntos imediatos,  $\{a, b, d\}$ ,  $\{a, c, d\}$  e  $\{a, d, e\}$ , são infrequentes. Por outro lado,  $\{a, c\}$  não pode ser considerado máximo porque um de seus superconjuntos imediatos,  $\{a, c, e\}$ , é frequente.

Figura 17 – Conjuntos de itens frequentes máximos.



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

Os conjuntos de itens frequentes máximos representam o menor grupo de conjuntos de itens responsável por gerar todos os outros conjuntos de itens frequentes. Por exemplo, os conjuntos de itens frequentes demonstrados na Figura 17 podem ser divididos em dois grupos:

- Conjuntos de itens frequentes iniciados pelo item  $a$  e que podem conter itens  $c$ ,  $d$  ou  $e$ . Esse grupo inclui conjuntos como  $\{a\}$ ,  $\{a, c\}$ ,  $\{a, d\}$ ,  $\{a, e\}$  e  $\{a, c, e\}$ ; e
- Conjuntos de itens frequentes iniciados por itens  $b$ ,  $c$ ,  $d$  ou  $e$ . Esse grupo inclui conjuntos como  $\{b\}$ ,  $\{b, c\}$ ,  $\{c, d\}$ ,  $\{b, c, d, e\}$ , dentre outros.

Conjuntos de itens frequentes pertencentes ao primeiro grupo são subconjuntos de  $\{a, c, e\}$  ou  $\{a, d\}$ , enquanto os pertencentes ao segundo grupo são subconjuntos de  $\{b, c, d, e\}$ . Portanto, os conjuntos de itens frequentes máximos  $\{a, c, e\}$ ,  $\{a, d\}$  e  $\{b, c, d, e\}$  formam uma representação compacta dos conjuntos de itens frequentes demonstrados na Figura 17.

Pode-se notar que os conjuntos de itens frequentes máximos fornecem representações plausíveis para coleções de dados que podem produzir muitos conjuntos de itens frequentes, dado que a quantidade desses conjuntos está contida em ordens exponenciais. Entretanto, essa abordagem é prática apenas quando um algoritmo eficiente é utilizado para encontrar os conjuntos de itens frequentes máximos sem a necessidade de enumerar todos os seus

subconjuntos.

Embora esse método proporcione uma representação compacta dos dados, os conjuntos de itens frequentes máximos não possuem informações do suporte de seus subconjuntos. Portanto, é necessária uma nova iteração sobre o conjunto de dados, a qual será utilizada para determinar a contagem de suporte dos conjuntos de itens frequentes não máximos. Na próxima seção, é apresentada uma outra representação para os conjuntos de itens frequentes ([ZAKI, 2000](#)), a qual irá fornecer justamente a informação de suporte perdida nesse último caso.

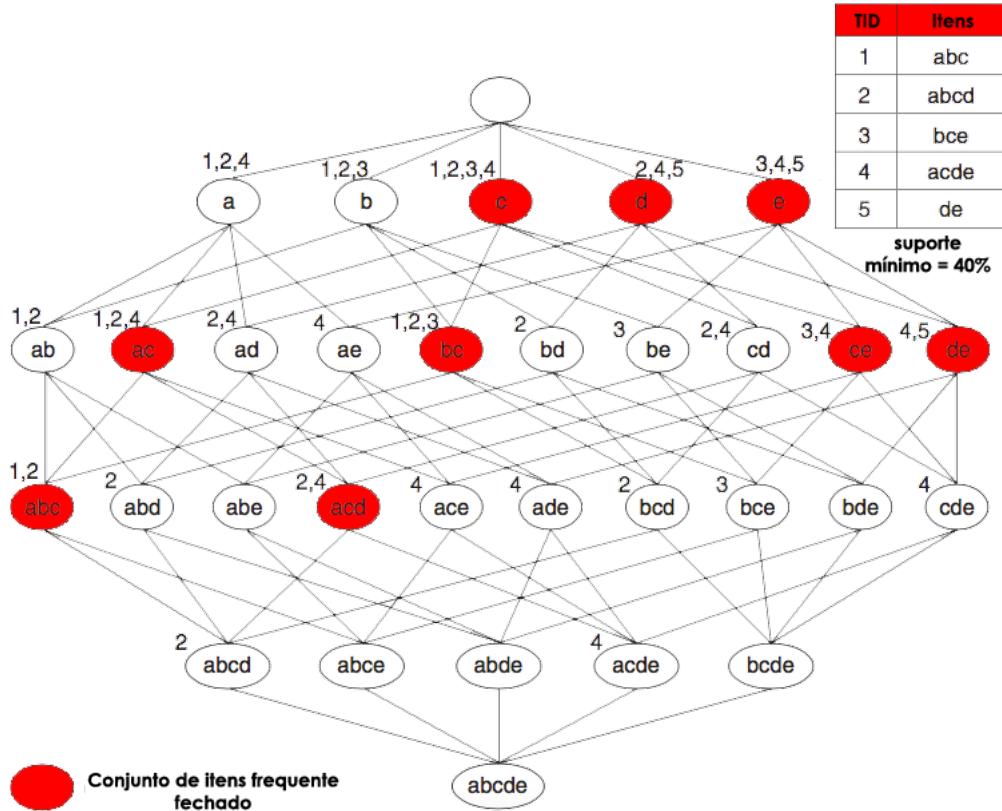
### **Conjuntos de itens frequentes fechado**

Um conjunto de itens fechado provém uma representação mínima dos conjuntos de itens sem a perda de suas informações de suporte. Um conjunto de itens  $X$  é fechado caso nenhum de seus superconjuntos imediatos possua a mesma contagem de suporte que  $X$ . Em outras palavras,  $X$  não é fechado se pelo menos um único superconjunto imediato seu tenha a mesma contagem de suporte. A Figura 18 ilustra melhor os exemplos de conjuntos de itens fechados. Cada nó da estrutura em grade está associado com as IDs das transações que contém esse nó, por exemplo, o nó  $\{b, c\}$  está associado às transações 1, 2 e 3, e sua contagem de suporte é equivalente à três. Dado esse diagrama, note que o suporte do conjunto  $\{b\}$  é idêntico ao de  $\{b, c\}$ ; portanto  $\{b\}$  não poderá ser considerado um conjunto de itens fechado. Por outro lado,  $\{b, c\}$  é um conjunto de itens fechado porque não possui a mesma contagem de suporte que qualquer um de seus superconjuntos.

Para o problema em questão, deve-se considerar os conjuntos de itens frequentes fechados, onde sua única diferença para o anterior é que o suporte deve ser maior ou igual ao *minsupport* utilizado na mineração. No exemplo anterior assumimos que o limite mínimo de suporte era de 40%, fato que garante o conjunto  $\{b, c\}$ , com suporte igual a 60%, como frequente e fechado.

É possível utilizar os conjuntos de itens frequentes fechados para determinar a contagem de suporte para os conjuntos de itens frequentes não fechados. Por exemplo, considere o conjunto de itens frequente  $\{a, d\}$ . Devido ao fato desse conjunto de itens não ser fechado, a sua contagem de suporte deve ser idêntica à um de seus superconjuntos imediatos. A solução é determinar quais superconjuntos ( $\{a, b, d\}$ ,  $\{a, c, d\}$  ou  $\{a, d, e\}$ ) possuem exatamente a mesma contagem de suporte que  $\{a, d\}$ . O princípio *Apriori* estabelece que qualquer transação que contenha o superconjunto de  $\{a, d\}$  também deve conter o conjunto  $\{a, d\}$ . Entretanto, qualquer transação que contenha  $\{a, d\}$  não necessita possuir os superconjuntos de  $\{a, d\}$ . Logo, o suporte de  $\{a, d\}$  deve ser igual ao maior suporte dentre os seus superconjuntos. Dado que  $\{a, c, d\}$  tem um suporte maior que  $\{a, b, d\}$  e  $\{a, d, e\}$ , o suporte de  $\{a, d\}$  deve ser idêntico ao suporte de  $\{a, c, d\}$ . A partir dessa metodologia, é possível criar um algoritmo para calcular o suporte dos conjuntos de itens frequentes não fechados. O Algoritmo 2 baseia-se numa abordagem específica x geral, ou seja, ele parte dos maiores conjuntos de itens frequentes para os menores. Isso acontece pois, para encontrar o suporte de um conjunto de itens frequente

Figura 18 – Conjuntos de itens frequentes fechados com suporte mínimo de 40%.



Fonte: Elaborada pelo autor.

não fechado, é necessário possuir o suporte de todos os seus superconjuntos.

### Algoritmo 2 – CONTAGEM DE SUPORTE UTILIZANDO CONJUNTOS DE ITENS FREQUENTES FECHADOS

1. *Seja C o conjunto de itens frequentes fechado.*
2. *Seja  $k_{max}$  o tamanho máximo dos conjuntos de itens frequentes fechados.*
3.  $F_{k_{max}} = \{f \mid f \in C, |f| = k_{max}\}.$
4. **Para**  $k = k_{max} - 1, k \geq 1$ , **faça**
5.      $F_k = \{f \mid f \subset F_{k+1}, |f| = k\}.$
6.     **Para** cada  $f \in F_k$  **faça**
7.         **Se**  $f \notin C$  **então**
8.              $f.\text{suporte} = \max\{f'.\text{suporte} \mid f' \in F_{k+1}, f \subset f'\}.$
- 9.
- 10.

A utilização de conjuntos de itens frequentes fechados é útil para a remoção de regras

de associações redundantes. Uma regra de associação  $X \rightarrow Y$  é redundante se existe alguma outra regra  $X' \rightarrow Y'$  onde  $X$  seja um subconjunto de  $X'$  e  $Y$  um subconjunto de  $Y'$  tal que o suporte e a confiança de ambas as regras sejam idênticas. No exemplo demonstrado na Figura 18, o conjunto  $\{b\}$  não é frequente e fechado, enquanto  $\{b, c\}$  é fechado. Consequentemente, a regra de associação  $\{b\} \rightarrow \{d, e\}$  é redundante porque ela possui o mesmo suporte e confiança que  $\{b, c\} \rightarrow \{d, e\}$ . Tais regras redundantes não são geradas se forem utilizados conjuntos de itens frequentes fechados para a geração de regras. Finalmente, note que todos os conjuntos de itens frequentes máximos são fechados porque nenhum deles pode ter a mesma contagem de suporte que seus superconjuntos imediatos.

Figura 19 – Diagrama de Venn dos conjuntos de itens frequentes.



Fonte: Elaborada pelo autor.

#### 2.2.4 Algoritmo *FP-growth*

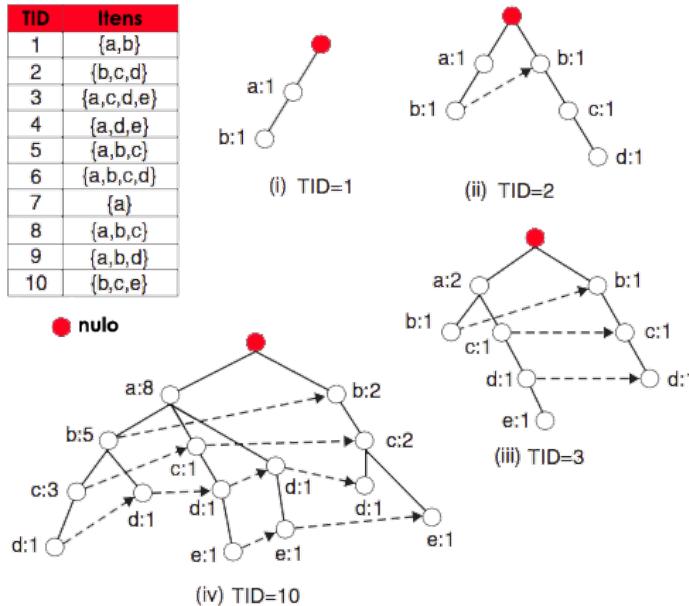
A diversidade dos problemas inerentes às tarefas efetuadas pela mineração de dados acabou por motivar a criação de diferentes abordagens para a execução desse processo. O algoritmo chamado FP-growth (HAN; PEI; YIN, 2000), ao contrário ao seu concorrente *Apriori*, utiliza uma metodologia totalmente diferente para a descoberta dos conjuntos de itens frequentes. A base de dados é codificada a partir do uso de uma estrutura de dados denominada árvore FP, onde os conjuntos de itens frequentes são extraídos diretamente desta estrutura.

##### **Árvore FP**

Uma árvore FP é uma representação compacta dos dados de entrada. Ela é construída a partir da leitura do conjunto de dados, onde cada transação será lida e mapeada, uma a uma, para um determinado caminho da árvore FP. Como diferentes transações podem possuir itens em comum, é possível que seus caminhos sejam sobrepostos. A cada caminho que é sobreposto, um nível maior de compressão é atingido utilizando a estrutura da árvore FP, possibilitando a

extração de conjuntos de itens frequentes diretamente da estrutura contida na memória ao invés de serem efetuadas diversas iterações de leitura sob os dados armazenados em disco.

Figura 20 – Construção de uma árvore FP.



Fonte: ([TAN; STEINBACH; KUMAR, 2005](#)). Adaptada pelo autor.

A Figura 20 mostra um conjunto de dados contendo dez transações e cinco itens. Cada nó da árvore contém o rótulo de um item, bem como um contador que identifica o número de transações mapeadas no caminho observado. Inicialmente, a árvore FP contém apenas o nó raiz, o qual é representado pelo símbolo de *nulo*. Posteriormente, essa árvore é estendida como segue:

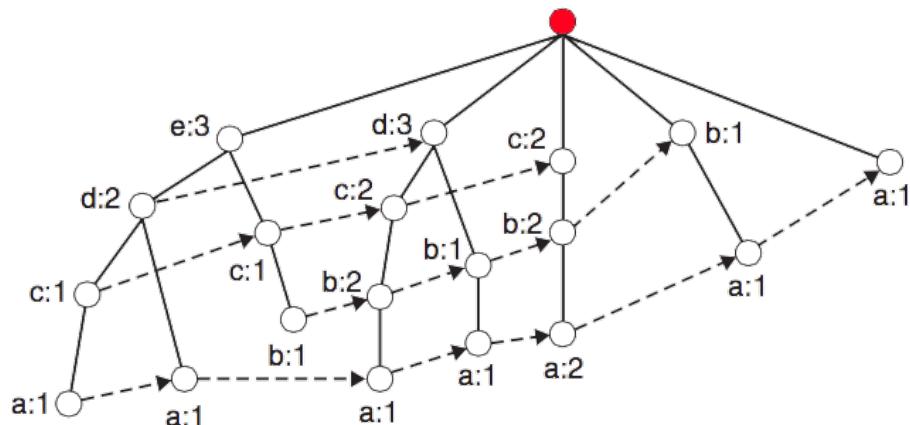
- A base de dados é lida uma vez para determinar a contagem de suporte de cada item, onde itens infrequentes serão descartados enquanto itens frequentes serão ordenados por ordem decrescente de contagem de suporte. Para os dados demonstrados na Figura 20, o conjunto  $\{a\}$  é o item mais frequente, seguido por  $\{b\}$ ,  $\{c\}$ ,  $\{d\}$  e  $\{e\}$ ;
- O algoritmo efetua uma segunda iteração sobre os dados para construir a árvore FP. Após a leitura da primeira transação,  $\{a, b\}$ , os nós rotulados por  $a$  e  $b$  são criados. Um caminho é formado a partir do *nulo*  $\rightarrow a \rightarrow b$  para codificar a transação. Cada nó contido nesse caminho tem uma contagem de frequência de valor igual a um;
- Após a leitura da segunda transação,  $\{b, c, d\}$ , um novo conjunto de nós é criado para os itens  $\{b\}$ ,  $\{c\}$  e  $\{d\}$ . Logo, um novo caminho é formado a fim de representar essa nova transação conectando os nós *nulo*  $\rightarrow b \rightarrow c \rightarrow d$ . Cada nó desse caminho também possui uma contagem de frequência igual a um. Embora as duas primeiras

transações possuam um item em comum,  $b$ , os seus caminhos são disjuntos pois elas não possuem um prefixo em comum;

- d) A terceira transação,  $\{a, c, d, e\}$ , possui um item prefixo em comum ( $a$ ) com a primeira transação. Por conseguinte, o caminho para esta transação  $nulo \rightarrow a \rightarrow c \rightarrow d \rightarrow e$  sobrepõe o caminho da primeira transação  $nulo \rightarrow a \rightarrow b$ . Por conta dessa sobreposição, a contagem de frequência do nó  $a$  é incrementada para dois enquanto as contagens de frequência para os nós recém criados,  $c, d$  e  $e$ , são iguais a um; e
- e) Esse procedimento continua até que todas as transações sejam mapeadas em um dos caminhos dados pela árvore FP. A árvore FP resultante desse processo é demonstrada no final da Figura 20.

Geralmente, o tamanho de uma árvore FP é menor do que o tamanho dos dados não comprimidos, principalmente por conta do fato das transações de mercado possuírem diversos itens em comum entre si. No melhor cenário possível, onde todas as transações possuem o mesmo conjunto de itens, a árvore FP contém apenas um único ramo de nós. Entretanto, o pior caso ocorre quando cada transação possui um conjunto único de itens. Como essa estrutura é mais complexa, cheia de ponteiros e nós a serem guardados, ela acaba por se tornar menos eficiente neste caso, acarretando em um armazenamento maior do que os próprios dados originais.

Figura 21 – Construção de uma árvore FP utilizando um diferente método de ordenação do suporte de seus itens.



Fonte: ([TAN; STEINBACH; KUMAR, 2005](#)). Adaptada pelo autor.

O tamanho de uma árvore FP também depende da forma como os seus itens estão ordenados. Nem sempre uma ordenação de suporte na forma decrescente resultará na menor árvore possível e vice-versa. Uma árvore FP também contém uma lista de ponteiros que conectam nós que possuem os mesmos itens. Esses ponteiros, representados pelas linhas pontilhadas nas Figuras 20 e 21, facilitam o rápido acesso aos itens individuais contidos na árvore.

### Geração de conjuntos de itens frequentes no algoritmo *FP-growth*

O algoritmo *FP-growth* é responsável por gerar conjuntos de itens frequentes de uma árvore FP a partir de uma metodologia de exploração ascendente, ou seja, o algoritmo primeiramente procura por conjuntos de itens frequentes que terminam em *e*, seguidos por *d*, *c*, *b* e finalmente, *a*. Essa abordagem é equivalente ao método baseado em sufixo demonstrado na Seção 2.2.5. Dado que cada transação é mapeada para um caminho na árvore FP, pode-se derivar os conjuntos de itens frequentes que terminam em itens particulares. Por exemplo, para o item *e* serão examinados apenas os caminhos contendo o nó *e*. Como dito anteriormente, esses caminhos podem ser acessados rapidamente utilizando os ponteiros associados ao nó *e*.

Após encontrar os conjuntos de itens frequentes terminados em *e*, o algoritmo procura os conjuntos de itens frequentes terminados em *d*. Esse processo continua até que todos os caminhos associados aos nós restantes, *c*, *b* e *a*, sejam processados. A Tabela 3 ilustra os conjuntos de itens frequentes correspondentes aos seus nós.

Tabela 3 – Lista dos conjuntos de itens frequentes correspondentes aos seus sufixos.

Sufixo	Conjuntos de itens frequentes
<i>e</i>	{e}, {d, e}, {a, d, e}, {c, e}, {a, e}
<i>d</i>	{d}, {c, d}, {b, c, d}, {a, c, d}, {b, d}, {a, b, d}, {a, d}
<i>c</i>	{c}, {b, c}, {a, b, c}, {a, c}
<i>b</i>	{b}, {a, b}
<i>a</i>	{a}

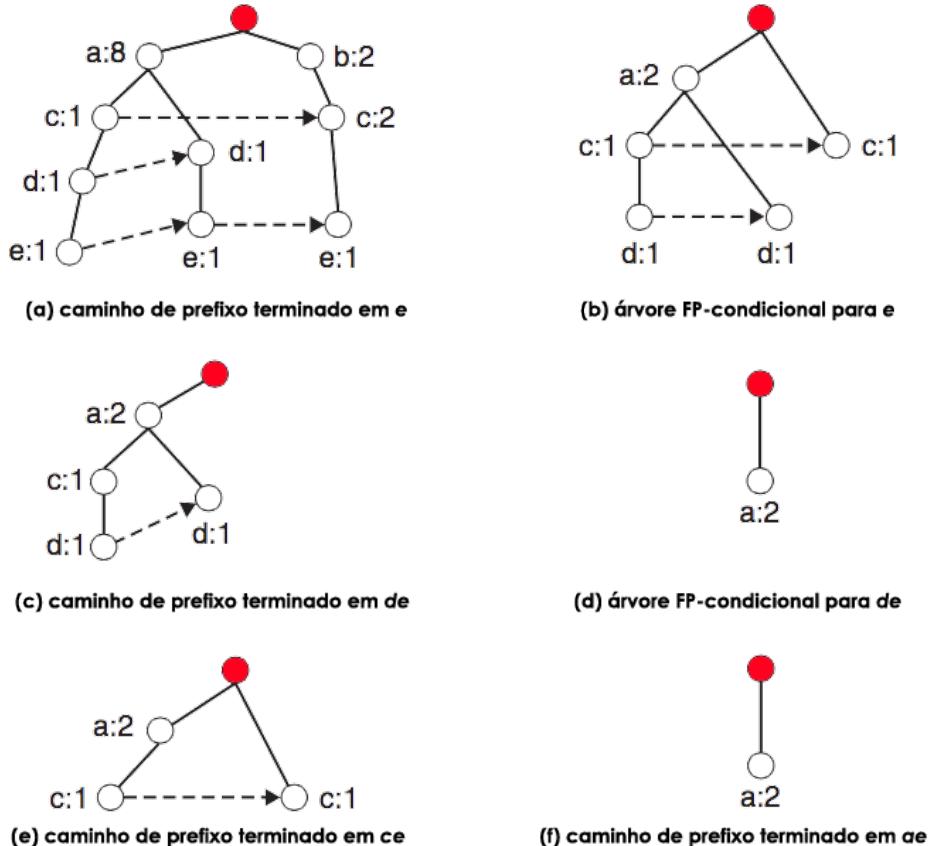
Fonte: Elaborada pelo autor.

O *FP-growth* utiliza uma estratégia de divisão e conquista para efetuar esse procedimento, onde o problema inicial é dividido em subproblemas menores. Por exemplo, para encontrar todos os conjuntos de itens frequentes terminados em *e*, primeiramente é necessário examinar se o próprio conjunto {*e*} é frequente. Em caso positivo, os subproblemas inerentes à esse nó, *de*, *ce*, *be* e *ae*, são considerados e examinados. A partir da associação das soluções obtidas pelos subproblemas todos os conjuntos de itens frequentes terminados em *e* podem ser encontrados. Essa abordagem de divisão e conquista é a peça chave da metodologia empregada pelo algoritmo *FP-growth*.

Para um melhor entendimento dessa metodologia, será descrito a seguir um passo-a-passo da descoberta dos conjuntos de itens frequentes terminados pelo item *e*:

- Todos os caminhos contendo o nó *e* são postos em questão. Esses caminhos iniciais são chamados de **caminhos de prefixo**;
- A partir desses caminhos de prefixo, a contagem de suporte de *e* é obtida a partir da adição das contagens de suporte associadas ao nó *e*. Seja o mínimo suporte igual a dois. Portanto, o conjunto {*e*} com contagem de suporte igual a três pode ser declarado como um conjunto de itens frequente;

Figura 22 – Aplicação do algoritmo *FP-growth* para encontrar os conjuntos de itens frequentes terminados em *e*.



Fonte: ([TAN; STEINBACH; KUMAR, 2005](#)). Adaptada pelo autor.

- c) Dado que  $\{e\}$  seja frequente, o algoritmo deve resolver os subproblemas inerentes à procura dos conjuntos de itens frequente terminados em *de*, *ce*, *be* e *ae*. Entretanto, é necessário converter os caminhos de prefixo em árvores FP condicionais, as quais possuem estruturas similares à uma árvore FP, exceto pelo fato de que são usadas para encontrar conjuntos de itens frequentes que terminam em determinados sufixos. Essa árvore condicional é obtida como segue:
- As contagens de suporte ao longo dos caminhos de prefixo devem ser atualizadas, pois algumas contagens incluem transações que não contém o item *e*. Por exemplo, o caminho *nulo*  $\rightarrow b : 2 \rightarrow c : 2 \rightarrow e : 1$  inclui a transação  $\{b, c\}$ , a qual não contém o item *e*. As contagens ao longo do caminho de prefixo devem ser ajustadas para 1, em reflexo ao número atual de transações contendo  $\{b, c, e\}$ ;
  - Os caminhos de prefixo são truncados ao remover os nós para *e*. Esses nós podem ser removidos pois as contagens de suporte ao longo dos caminhos de prefixo foram atualizadas apenas para as transações que contém *e*, não necessitando mais da informação desse nó *e* para os subproblemas terminados em *de*, *ce*, *be* e *ae*;

- Após essa atualização, alguns dos itens podem não ser mais frequentes. Por exemplo, o nó  $b$  aparece uma única vez, o que significa que existe apenas uma transação que contém ambos nós  $b$  e  $e$ . Portanto, o item  $b$  pode ser ignorado nas futuras análises porque todos os conjuntos de itens terminados em  $be$  serão infrequentes.

O processo de criação da árvore FP-condicional para  $e$  está demonstrada nas Figuras 22(a) e 22(b). A árvore difere de sua original pois as contagens de frequência foram atualizadas e os nós  $b$  e  $e$  foram removidos;

- A árvore FP-condicional para o nó  $e$  é utilizada para solucionar os subproblemas de encontrar os conjuntos de itens frequentes terminados em  $de$ ,  $ce$  e  $ae$ . A mesma abordagem para a construção da árvore FP-condicional de  $e$  é usada para a construção da árvore FP-condicional de  $de$ , ilustrada pela Figuras 22(c) e 22(d). Dado que essa árvore possui apenas um item  $a$ , que possui o valor mínimo de suporte estabelecido para o problema, o algoritmo descobre o conjunto frequente  $\{a, d, e\}$  e segue para a resolução do próximos subproblemas,  $ce$  e  $ae$ , ilustrados pelas Figuras 22(e) e 22(f), respectivamente.

Por meio de um processo recursivo, a árvore FP-condicional é construída pela atualização da contagem de frequência ao longo dos caminhos de prefixo e pela remoção de todos os itens infrequentes. Como os subproblemas são disjuntos, esse algoritmo não gerará conjuntos de itens duplicados. Além disso, para algumas bases de dados transacionais, o algoritmo *FP-growth* possui uma performance bastante superior ao tradicional algoritmo *Apriori*. O único problema remete-se à como estão estruturadas as árvores FP-condicionais pois, se estiverem muito populadas, o algoritmo levará um tempo maior para gerar um grande número de subproblemas e posteriormente combiná-los.

## 2.2.5 Métodos alternativos para geração de conjuntos de itens frequentes

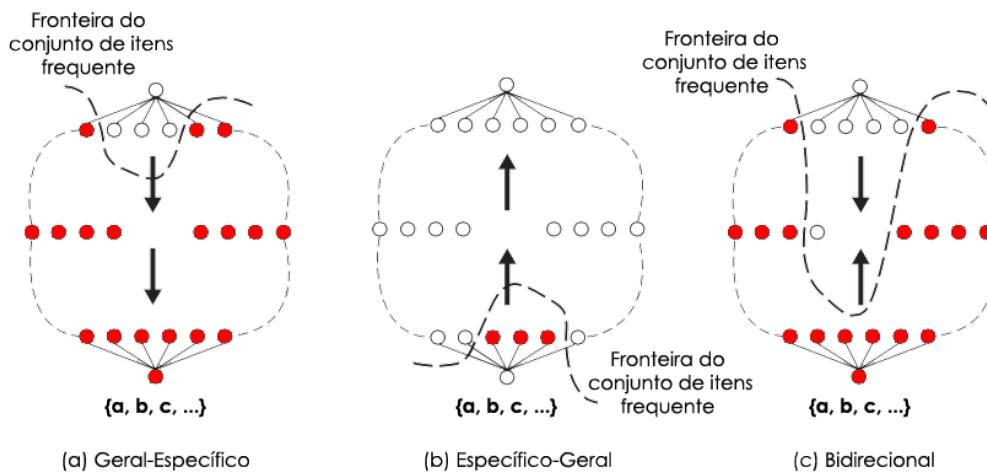
A técnica *Apriori* é um dos primeiros algoritmos que obteve sucesso em identificar e regular a expansão combinatorial da geração de conjuntos de itens frequentes. O princípio do algoritmo é aplicado para fazer o corte necessário no espaço de busca exponencial reduzindo o seu tamanho. Entretanto, o algoritmo ainda sofre bastante problemas de E/S devido ao fato de serem necessárias diversas iterações ao longo da base de dados transacional. Adicionalmente, a eficácia do *Apriori* pode se degradar razoavelmente quando confrontado com bases de dados muito densas, puramente por conta do aumento das informações contidas nas transações. Diversos outros métodos foram desenvolvidos com o intuito de contornar esse problema e melhorar a eficiência desse algoritmo. A seguir apresentamos alguns desses métodos para agregar um maior conhecimento ao leitor.

Uma busca ao longo da base à procura de conjuntos de itens frequentes pode também

ser vista como uma travessia ao longo da estrutura em grade do conjunto de itens, como demonstrado anteriormente pela Figura 7. A estratégia de busca empregada pelo algoritmo impõe como a estrutura deve ser percorrida durante o processo de geração dos conjuntos de itens frequentes. Algumas políticas de busca podem ser consideradas melhores que outras, dependendo do modo com que o conjunto de itens frequente está configurado na grade. A seguir algumas delas são apresentadas:

- a) **Geral-Específico x Específico-Geral:** O algoritmo *Apriori* utiliza uma estratégia de busca baseada no conceito de geral-específico, onde pares de conjuntos de  $(k - 1)$  itens frequentes são combinados para obter os conjuntos de  $k$  itens candidatos. Esse método de busca é efetivo, contanto que o comprimento máximo do conjunto de itens frequente não seja muito longo. A configuração de conjuntos de itens frequentes que melhor funcionam com essa estratégia está demonstrado na Figura 23(a), onde os nós escuros representam os conjuntos infrequentes. Alternativamente, uma estratégia de busca específico-geral atenta-se primeiramente aos conjuntos de itens frequentes mais específicos para depois encontrar os conjuntos mais gerais. Essa metodologia é usada para procurar os conjuntos de itens frequentes máximos em transações densas (LIN; KEDEM, 1998), onde a fronteira do conjunto de itens frequente está próxima da base da grade, como demonstrado na Figura 23(b).

Figura 23 – Diferentes tipos de busca para procura de conjuntos de itens frequentes.



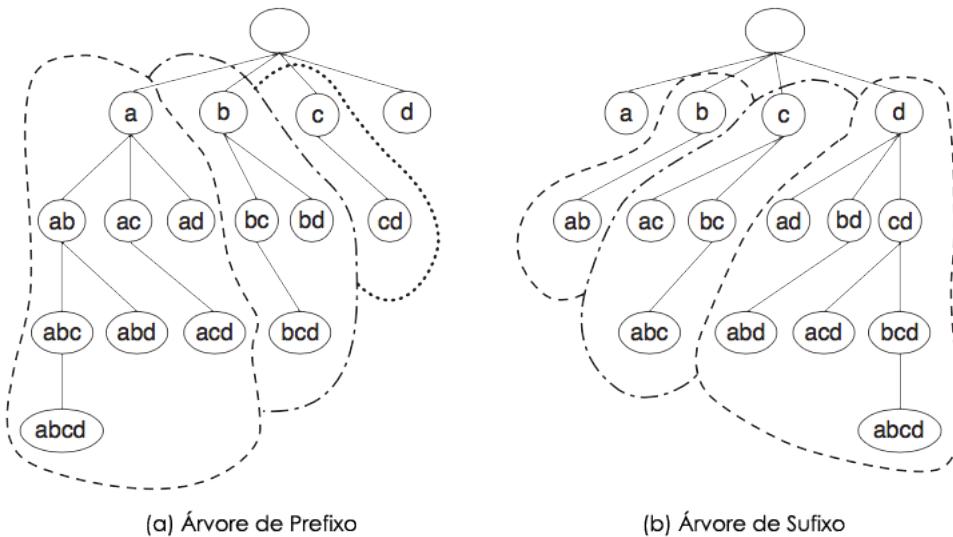
Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

O princípio *Apriori* também pode ser aplicado para cortar todos os subconjuntos dos conjuntos de itens frequentes máximos. Especificamente, se um conjunto de  $k$  itens candidato for frequente e máximo, não é necessário examinar nenhum de seus subconjuntos de tamanho  $(k - 1)$ . Entretanto, se o conjunto de  $k$  itens candidato for infrequente, é necessário checar todos os seus  $(k - 1)$  subconjuntos na próxima iteração. Outra abordagem é combinar as duas estratégias de busca. Essa visão

bidirecional requer um espaço maior para guardar os conjuntos de itens candidatos, mas auxilia rapidamente à identificar a fronteira do conjunto de itens frequente, demonstrado pela Figura 23(c);

- b) **Classes Equivalentes:** Outra maneira de visualizar a travessia é partitionar a grade em grupos disjuntos de nós (classes equivalentes). Um algoritmo de geração de conjunto de itens frequentes efetua primeiramente sua busca à procura de conjuntos de itens frequentes dentro de uma classe equivalente em particular para depois efetuar a busca em outra classe equivalente. Por exemplo, o algoritmo *Apriori* partitiona a grade em conjuntos de itens de mesmo tamanho a cada iteração, ou seja, ele primeiro encontra todos os conjuntos de  $k$  itens antes de prosseguir à procura dos conjuntos de  $(k + 1)$  itens.

Figura 24 – Tipos de árvores utilizadas na metodologia das classes equivalentes.



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

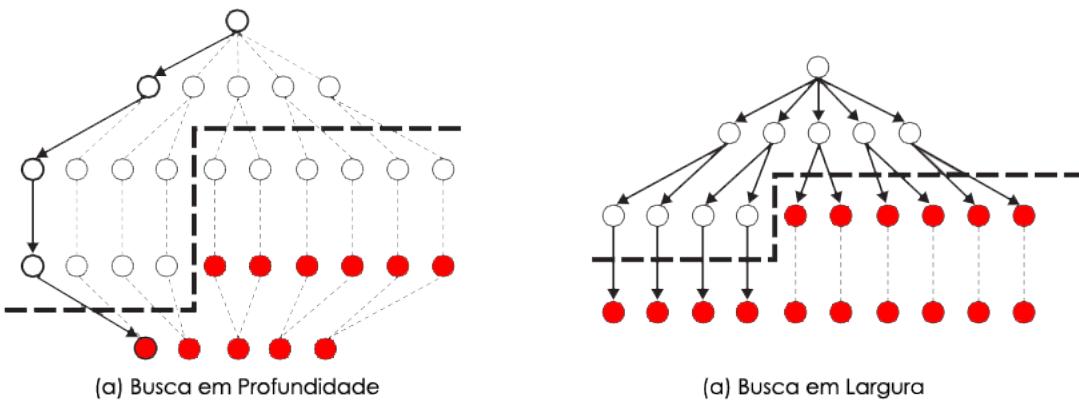
As classes equivalentes também podem ser definidas de acordo com os rótulos de prefixo e sufixo de um conjunto de itens. Neste caso, dois conjuntos de itens pertencem à mesma classe caso possuam em comum um prefixo ou sufixo de tamanho  $k$ . Na abordagem baseada em prefixo, o algoritmo sai em busca de conjuntos de itens frequentes que começam com o prefixo  $a$  antes de começar a procurar aqueles com prefixos  $b$ ,  $c$ , etc. As duas abordagens para as classes de equivalência, baseadas em prefixo e sufixo, estão ilustradas pela estrutura de árvore da Figura 24;

- c) **Largura x Profundidade:** O algoritmo *Apriori* percorre a grade por meio de uma busca em largura. Todos os conjuntos de um item frequentes são encontrados, seguidos pelos conjuntos de dois itens e assim por diante, até que não existam mais conjuntos de itens frequentes para serem gerados. Porém, a grade também

pode ser percorrida por uma busca em profundidade. O algoritmo pode começar, por exemplo, a partir do nó  $a$  e depois contar o seu suporte para determinar se ele é frequente ou não. Em caso positivo, o algoritmo continua progressivamente a expansão nos próximos níveis, por exemplo,  $ab$ ,  $abc$  e assim por diante, até que um nó infrequente seja alcançado, digamos  $abcd$ . Após o encontro desse nó, ele efetua uma operação de *backtrack*, ou seja, recua e avança para outro ramo, por exemplo  $abce$ , e continua o seu processo de busca a partir desse ponto.

A abordagem de busca em profundidade é ocasionalmente utilizada por algoritmos para a descoberta de conjuntos de itens frequentes máximos ([BURDICK; CALIMLIM; GEHRKE, 2001](#)). Esse método permite com que a fronteira do conjunto de item frequente seja detectada mais rapidamente do que utilizando a busca em largura. Quando o conjunto de itens frequente e máximo é encontrado, um corte é efetuado em seus subconjuntos. Por exemplo, se o nó  $bcd$  for frequente e máximo, então o algoritmo não precisa visitar as sub-árvores abaixo dos nós  $bd$ ,  $be$ ,  $c$ ,  $d$  e  $e$ , pois não existirão conjuntos de itens frequentes e máximos. Entretanto, se  $abc$  for frequente e máximo, apenas nós como  $ac$  e  $bc$  não serão frequentes e máximos (as sub-árvores de  $ac$  e  $bc$  ainda podem conter conjuntos de itens máximos e frequentes). A abordagem em profundidade também permite um diferente tipo de corte baseado no suporte dos conjuntos de itens. Por exemplo, suponha que o suporte de  $\{a, b, c\}$  seja idêntico ao de  $\{a, b\}$ . As sub-árvores abaixo dos nós  $abd$  e  $abe$  podem ser puladas pois há garantias de que não existirão conjuntos de itens frequentes e máximos.

Figura 25 – Exemplos de busca em profundidade e largura.



Fonte: ([TAN; STEINBACH; KUMAR, 2005](#)). Traduzida e adaptada pelo autor.

Embora existam diversos métodos para se efetuar a busca desses conjuntos, também é de extrema importância analisar a estrutura com que os dados transacionais estão sendo representados. A escolha da estrutura pode afetar os custos de E/S inerentes ao cálculo do suporte de conjuntos de itens candidatos. A Figura 26 é responsável por ilustrar duas diferentes maneiras de se representar as transações de mercado.

Figura 26 – Exemplos dos possíveis layouts de estrutura para armazenamento dos dados.

Layout Horizontal		Layout Vertical				
TID	Itens	a	b	c	d	e
1	a,b,e	1	1	2	2	1
2	b,c,d	4	2	3	4	3
3	c,e	5	5	4	5	6
4	a,c,d	6	7	8	9	
5	a,b,c,d	7	8	9		
6	a,e	8	10			
7	a,b	9				
8	a,b,c					
9	a,c,d					
10	b					

Fonte: Elaborada pelo autor.

A representação mais à esquerda é chamada de layout **horizontal** de dados, o qual é adotado por diversos algoritmos de mineração de regras de associação incluindo a implementação do algoritmo *Apriori* efetuada por Borgelt ([BORGELT, 2003](#)), a qual será uma das bibliotecas utilizadas nos experimentos desse trabalho. Outra possibilidade é armazenar a lista de identificadores das transações (lista de TIDs) associada com cada item. Tal representação é conhecida por layout **vertical** de dados. O suporte de cada conjunto de itens candidato é obtido através da intersecção das listas de TIDs de seus subconjuntos de itens. O comprimento das listas de TIDs diminuem ao longo do uso de conjuntos de itens maiores. Entretanto, um problema para essa abordagem pode ser notado quando o conjunto inicial das listas de TIDs são muito grandes para serem encaixados na memória principal, requerendo técnicas mais sofisticadas para efetuar a compressão dessas listas.

## 2.2.6 Outros algoritmos geradores

O algoritmo AIS ([AGRAWAL; IMIELINSKI; SWAMI, 1993b](#)) gera conjuntos de itens candidatos em tempo real enquanto a base de dados está sendo lida. Especificamente, após ler uma transação, é determinado quais dos conjuntos de itens considerados frequentes no passo anterior estão presentes na transação. Novos conjuntos de itens candidatos são gerados extendendo esses conjuntos de itens frequentes com outros itens durante a transação. Contudo, isto é um processo desvantajoso, dado que serão gerados resultados desnecessários e muitos conjuntos candidatos pequenos serão contados.

Os algoritmos Apriori e AprioriTid ([AGRAWAL; SRIKANT, 1994](#)) geram conjuntos de itens candidatos apenas utilizando os conjuntos de itens definidos como frequentes pela iteração anterior, sem considerar as transações existentes na base de dados. A intuição básica

é que qualquer subconjunto de um conjunto de itens frequente deve ser frequente. Portanto, os conjuntos de itens candidatos contendo  $k$  itens podem ser gerados a partir da junção de conjuntos de itens frequentes que contenham  $(k - 1)$  itens, deletando todos que possuam um subconjunto que não é considerado frequente. Esse procedimento resulta na geração de um número bem menor de conjuntos candidatos.

O algoritmo AprioriTid ainda possui a propriedade de que a base de dados não é totalmente utilizada na contagem do suporte dos conjuntos de itens candidatos após a primeira iteração. Preferivelmente, uma codificação dos conjuntos candidatos utilizados no passo anterior é empregada para esse propósito. Nos passos subsequentes, o tamanho dessa codificação pode ficar bem menor que a base de dados, salvando muito esforço de leitura.

### **AIS (Agrawal, Imielinski, Swami)**

A fundamentação teórica deste trabalho de conclusão de curso teve por estudo principal a obra de Agrawal “*Mining Association Rules between Sets of items in Large Databases*” ([AGRAWAL; IMIELINSKI; SWAMI, 1993b](#)), onde é introduzido o primeiro algoritmo para a mineração de dados por meio de regras de associação.

O algoritmo AIS efetua múltiplas iterações sobre a base de dados gerando um grupo de fronteira para cada iteração, o qual consiste nos conjuntos de itens que foram extendidos durante a operação. Em cada iteração, a contagem de suporte é efetuada para determinados conjuntos de itens. Esses conjuntos de itens, denominados de conjuntos de itens candidatos, são derivados das tuplas contidas na base de dados e dos conjuntos de itens contidos no grupo de fronteira. Também há um contador associado a cada conjunto de itens responsável por armazenar o número de transações em que o correspondente conjunto está presente. Lembrando que esse contador é inicializado em zero quando um novo conjunto de itens é criado.

Este algoritmo é capaz de gerar apenas regras de associação com um item no consequente, ou seja, somente são geradas regras como  $X \cap Y \rightarrow Z$ , mas não as regras como  $X \rightarrow Y \cap Z$ . A seguir, é apresentado o esboço inicial de seu algoritmo:

### Algoritmo 3 – ALGORITMO AIS

ENTRADA: Conjunto Frequent L = 0 e Grupo de Fronteira F = 0.

1. **Enquanto**  $F \neq 0$ , **faça**
2.     **Seja** conjunto candidato  $C = 0$ ;
3.     **Para cada** tuplas  $t$  da base de dados, **faça**
4.         **Para cada** conjuntos de itens  $f$  em  $F$ , **faça**
5.             **Se**  $t$  contém  $f$ , **então**
6.                 **Seja**  $C_f = \text{conjuntos de itens candidatos extensões de } f \text{ e contidos em } t$ ;
7.             **Para cada** conjunto de itens  $c_f$  em  $C_f$ , **faça**
8.                 **Se**  $c_f \in C$ , **então**
9.                      $c_f.\text{contagem} = c_f.\text{contagem} + 1$ ;
10.                 **Se não**
11.                      $c_f.\text{contagem} = 0$ ;
12.                  $C = C + c_f$ ;
- 13.
- 14.
- 15.
16.     **Seja**  $F = 0$ ;
17.     **Para cada** conjunto de itens  $c$  em  $C$ , **faça**
18.         **Se**  $\text{contagem}(c)/\text{tamanho\_db} > \text{minsupport}$ , **então**
19.              $L = L + c$ ;
20.         **Se**  $c$  deve ser usado como a próxima fronteira, **então**
21.              $F = F + c$ ;
- 22.
- 23.

Inicialmente, o grupo de fronteira consiste em apenas um elemento, representado pelo conjunto vazio. No final da iteração a contagem de suporte para um conjunto de itens candidato é efetuada e comparada ao *minsupport* para determinar se é ou não um conjunto de itens frequente. Ao mesmo tempo, também é determinado se o conjunto de itens deve ser adicionado ao grupo de fronteira para a próxima iteração, onde terá sua contagem de suporte preservada independente do resultado desta operação. O algoritmo termina quando o grupo de fronteira se torna vazio.

Exemplificando melhor o processo de geração dos conjuntos de itens candidatos, o algoritmo lê uma tupla da base de dados por vez e checa quais grupos de fronteira estão contidos na tupla lida. Conjuntos de itens candidatos são gerados a partir do conjunto de itens de fronteira ao extendê-los recursivamente com outros itens presentes na tupla. Um conjunto de itens considerado infrequente não deve sofrer uma nova extensão. Para que

não ocorra a construção dos mesmos conjuntos de itens de diferentes formas, os itens estão ordenados lexicograficamente e um conjunto de itens  $X$  é apenas extendido por itens ordenados posteriormente a qualquer membro de  $X$ .

**Algoritmo 4** – PROCEDIMENTO Extensão( $X$ : CONJUNTO DE ITENS,  $t$ : TUPLA)

1. **Seja** o item  $I_j$  tal que  $\forall I_t \in X, I_j \geq I_t$ ;
2. **Para todo** item  $I_k$  contido na tupla  $t$  tal que  $I_k > I_j$  **faça**
3.     **saída**( $XI_k$ );
4.     **Se** ( $XI_k$ ) é estimado como frequente, **então**
5.         **Extensão**( $XI_k$ ,  $t$ );
- 6.

Por exemplo, seja  $I$  um conjunto de itens igual a  $\{A, B, C, D, E, F\}$ , com os itens ordenados em ordem alfabética. Também assuma que o grupo de fronteira contenha apenas um conjunto de itens AB. Para a tupla contida na base de dados  $t = ABCDF$ , os seguintes conjuntos de itens candidatos são gerados:

- a) ABC estimado frequente: continuar a extensão;
- b) ABCD estimado infrequente: não continuar a extensão;
- c) ABCF estimado frequente: não pode mais ser extendido;
- d) ABD estimado infrequente: não continuar a extensão; e
- e) ABF estimado frequente: não pode mais ser extendido.

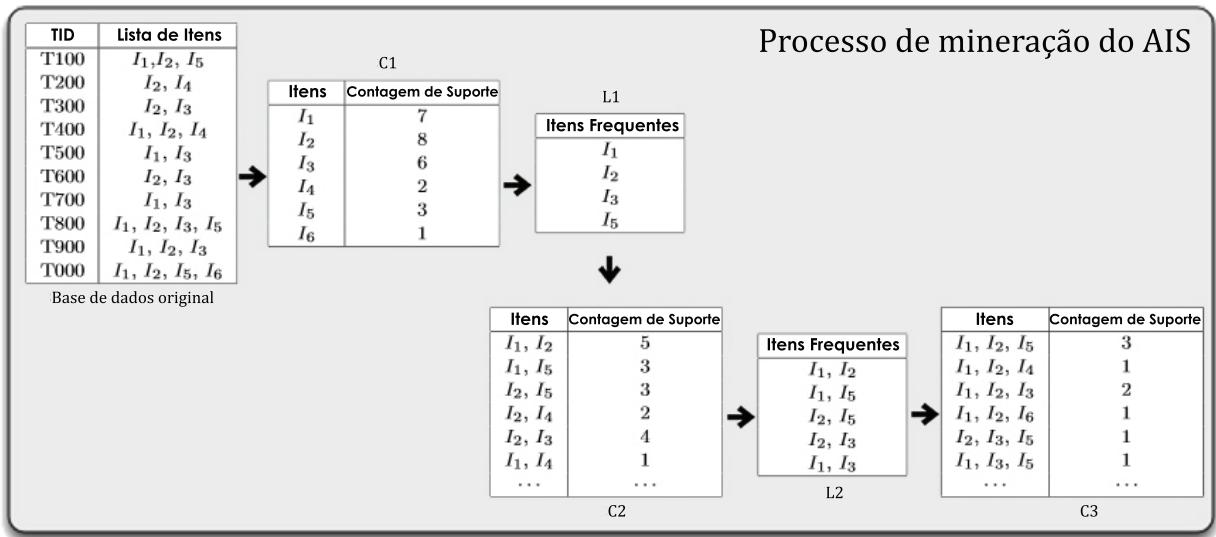
A extensão ABCDF não foi considerada, pois ABCD é previsto como um conjunto infrequente, ao passo que a extensão ABDF também não foi considerada porque ABD é previsto como infrequente. Os conjuntos de itens ABCF e ABF, mesmo tendo sido estimados como frequentes, não podem ser estendidos posteriormente, porque não há item na tupla  $t$  que seja maior que o item F.

Os conjuntos de itens candidatos estimados como infrequentes, mas descobertos como frequentes na atual iteração, são incluídos no grupo de fronteira. Para sustentar a hipótese que esses são os únicos conjuntos necessários a serem inclusos no próximo grupo de fronteira, o seguinte teorema é proposto por Agrawal et. al ([AGRAWAL; IMIELINSKI; SWAMI, 1993b](#)):

**Teorema:** Se o conjunto de itens candidato  $X$  é estimado como infrequente na atual iteração sobre a base de dados, então nenhuma extensão  $X + I_j$  de  $X$  é feita, onde  $I_j > I_k$  para qualquer  $I_k$  contido em  $X$ , tal que este seja um conjunto de itens candidato nesta iteração.

O teorema é verdadeiro devido ao procedimento de geração dos conjuntos de itens candidatos. Se um conjunto de itens candidato era frequente mas não foi estimado como infrequente, então ele não deve ser adicionado ao próximo grupo de fronteira, pois do modo como o algoritmo foi definido, todas as extensões de um conjunto de itens nesta situação já foram consideradas nesta iteração. Já um conjunto de itens candidato considerado infrequente, não deve ser incluído no próximo grupo de fronteira, pois o suporte de uma extensão do conjunto de itens não pode ser maior do que o suporte do próprio conjunto. Um pequeno exemplo do processo de mineração do algoritmo AIS está representado na Figura 27.

Figura 27 – Processo de mineração do algoritmo AIS.



Fonte: Elaborada pelo autor.

### AprioriTid

O algoritmo AprioriTid também faz o uso da função *apriorigen* para determinar os conjuntos de itens candidatos prévios ao início da iteração. O intrigante fato desse algoritmo é que a base de dados, por exemplo  $D$ , não é utilizada para a contagem de suporte após a primeira iteração; ao invés disso, é utilizado um conjunto  $\bar{C}_k$ . Cada membro do conjunto  $\bar{C}_k$  consiste no formato  $\langle TID, \{X_k\} \rangle$ , onde cada  $X_k$  é potencialmente um conjunto frequente de  $k$  itens presente na transação com identificador  $TID$ . Para  $k = 1$ ,  $\bar{C}_1$  corresponde a base de dados  $D$ , embora cada item  $i$  seja substituído pelo conjunto de itens  $\{i\}$ . Para  $k > 1$ ,  $\bar{C}_k$  é gerado pelo Algoritmo 5 (linha 11). Cada membro de  $\bar{C}_k$  correspondente à transação  $t$  é dado por  $\langle t.TID, \{c \in C_k \mid c \text{ está contido em } t\} \rangle$ . Se a transação não contém qualquer conjunto de  $k$  itens candidato, então  $\bar{C}_k$  não terá uma entrada para essa transação. Portanto, o número de entradas em  $\bar{C}_k$  pode ser menor do que o número de transações na base de dados, especialmente para altos valores de  $k$ . Adicionalmente, para altos valores de  $k$ , cada entrada pode ser menor do que a correspondente transação, porque apenas poucos candidatos

poderão estar contidos nessa transação. Entretanto, para pequenos valores de  $k$ , cada entrada pode ser maior do que a transação correspondente, porque uma entrada em  $C_k$  inclui todos os conjuntos de  $k$  itens candidatos contidos na transação.

### Algoritmo 5 – ALGORITMO APRIORITID

1.  $L_1 = \{\text{conjuntos de 1 item frequentes}\};$
2.  $\bar{C}_1 = \text{base de dados } D;$
3. **Para** ( $k = 2; L_{k-1} \neq 0; k++$ ) **faz**
4.      $C_k = \text{apriorigen}(L_{k-1});$
5.      $\bar{C}_k = 0;$
6.     **Para todas** entradas  $t \in \bar{C}_{k-1}$  **faz**
7.          $C_t = \{c \in C_k \mid (c - c[k]) \in t.\text{setofitemsets} \wedge (c - c[k-1]) \in t.\text{setofitemsets}\};$
8.         **Para todos** candidatos  $c \in C_t$  **faz**
9.              $c.\text{contagem} ++;$
10.         **Se** ( $C_t \neq 0$ ) **então**
11.              $\bar{C}_k += <t.TID, C_t>;$
12.      $L_k = \{c \in C_k \mid c.\text{contagem} \geq \text{minsupport}\};$
- 13.
14. **Resposta** =  $\bigcup_k L_k;$

Figura 28 – Algoritmo AprioriTid.

Base de dados		$\bar{C}_1$		$L_1$	
TID	Itens	TID	Grupo de conjunto de itens	Itens	Suporte
100	1 3 4	100	{ {1}, {3}, {4} }	{1}	2
200	2 3 5	200	{ {2}, {3}, {5} }	{2}	3
300	1 2 3 5	300	{ {1}, {2}, {3}, {5} }	{3}	3
400	2 5	400	{ {2}, {5} }	{5}	3

C2		$\bar{C}_2$		$L_2$	
Itens		TID	Grupo de conjunto de itens	Itens	Suporte
{1 2}		100	{ {1, 3} }	{1 3}	2
{1 3}		200	{ {2, 3}, {2, 5}, {3, 5} }	{2 3}	2
{1 5}		300	{ {1, 2}, {1, 3}, {1, 5}, {2, 3}, {2, 5}, {3, 5} }	{2 5}	3
{2 3}		400	{ {2, 5} }	{3 5}	2
{2 5}					
{3 5}					

C3		$\bar{C}_3$		$L_3$	
Itens		TID	Grupo de conjunto de itens	Itens	Suporte
{2 3 5}		200	{ {2, 3, 5} }	{2 3 5}	2
		300	{ {2, 3, 5} }		

Fonte: Elaborada pelo autor.

**Exemplo:** Considere a base de dados da Figura 28 e assuma que seu suporte mínimo seja de duas transações. A função *apriorigen* chamada com  $L_1$  resulta nos conjuntos de itens candidatos  $C_2$ . Após, é feita a contagem de suporte dos candidatos presentes em  $C_2$  a partir da iteração das entradas em  $\bar{C}_1$ , gerando  $\bar{C}_2$ . A primeira entrada em  $\bar{C}_1$  é  $\{ \{1\}, \{3\}, \{4\} \}$ , correspondente à transação 100. O conjunto  $C_t$  correspondente à essa entrada  $t$  é  $\{ \{1, 3\} \}$ , devido ao fato de  $\{1, 3\}$  ser membro de  $C_2$  e ambos ( $\{1, 3\} \setminus \{1\}$ ) e ( $\{1, 3\} \setminus \{3\}$ ) serem membros de  $t.setofitemsets$ .

Chamando a função *apriorigen* com  $L_2$  é retornado o conjunto  $C_3$ . Efetua-se uma iteração sobre os dados com  $\bar{C}_2$  e  $C_3$  para gerar o conjunto  $\bar{C}_3$ . Note que não há entradas de  $\bar{C}_3$  para as transações com TIDs 100 e 400, dado que elas não possuem nenhum dos conjuntos de itens de  $C_3$ . O conjunto candidato  $\{2, 3, 5\}$  em  $C_3$  é frequente e o único membro de  $L_3$ . Quando gerado  $C_4$  utilizando  $L_3$ , um conjunto vazio é construído e o algoritmo termina.

### AprioriHybrid

Os algoritmos *Apriori* e *AprioriTid* utilizam o mesmo procedimento de geração de conjuntos de candidatos, contabilizando os mesmos conjuntos de itens. Entretanto, o *Apriori* ainda examina todas as transações contidas na base de dados enquanto o *AprioriTid* analisa  $\bar{C}_k$  para obter a contagem de suporte, mantendo o tamanho de  $\bar{C}_k$  inferior ao da própria base de dados. Adicionalmente, o custo de escrita no disco não é contabilizado quando os conjuntos  $\bar{C}_k$  estão gravados diretamente na memória.

Baseando-se nessas observações, foi possível a criação de um algoritmo híbrido denominado *AprioriHybrid* (AGRAWAL; SRIKANT, 1994), o qual utiliza o algoritmo *Apriori* em seus passos iniciais e muda para os procedimentos do *AprioriTid* quando é esperado que o conjunto  $\bar{C}_k$ , ao final de sua iteração, seja de tamanho suficiente para caber diretamente na memória. Para estimar se  $\bar{C}_k$  caberá na memória, ao final da iteração atual tem-se a contagem dos candidatos em  $C_k$ . A partir disso, é estimado o tamanho de  $\bar{C}_k$  caso ele fosse gerado. Esse tamanho é dado por:

$$\sum_{\text{candidatos } c \in C_k} \text{suporte}(c) + \text{número de transações.}$$

Caso  $\bar{C}_k$  nessa iteração tenha sido pequeno o suficiente para caber na memória, e se foram produzidos menos candidatos frequentes do que na iteração anterior, então é utilizado o *AprioriTid*. Essa última condição refere-se à não permissão da troca de  $\bar{C}_k$ , pois não há certeza de que ele caberá na memória na próxima iteração.

Não obstante, a substituição do algoritmo *Apriori* pelo *AprioriTid* envolve um custo adicional. Assuma que essa substituição tenha sido efetuada ao final da iteração  $k$ . Na iteração  $(k+1)$  após a descoberta dos conjuntos de itens candidatos contidos na transação, também será necessária a adição dos seus IDs para  $\bar{C}_{k+1}$ . Consequentemente, há novamente um custo adicional relativo à execução do algoritmo *Apriori*, pois apenas na iteração  $(k+2)$  o algoritmo

AprioriTid irá ser executado. Portanto, se não há conjuntos de  $(k + 1)$  itens frequentes ou  $(k + 2)$  candidatos, o custo adicional da substituição será contabilizado sem nenhum dos benefícios do AprioriTid terem sido obtidos.

### 2.2.7 Geração de regras

Outra tarefa inerente da mineração de dados é a extração de regras eficientes a partir dos conjuntos de itens frequentes encontrados (BAYARDO JR.; AGRAWAL, 1999). Cada conjunto  $Y$  de  $k$  itens frequente pode produzir até  $2^k - 2$  regras de associação, número bastante grande se levarmos em conta que não existe a produção de regras que possuam seu antecedente ou consequente vazios, por exemplo,  $\emptyset \rightarrow Y$  ou  $Y \rightarrow \emptyset$ . Uma regra de associação pode ser extraída a partir do particionamento do conjunto de itens  $Y$  em dois subconjuntos não vazios,  $X$  e  $Y - X$ , tal que  $X \rightarrow Y - X$  satisfaça o limite mínimo de confiança. Note que todas essas regras já cumpriram com os requisitos mínimos de suporte, dado que elas foram geradas a partir de um conjunto de itens frequente.

**Exemplo 2:** Seja  $X = \{1, 2, 3\}$  um conjunto de itens frequente. Existem seis regras de associação candidatas que podem ser geradas a partir de  $X$ :  $\{1, 2\} \rightarrow \{3\}$ ,  $\{1, 3\} \rightarrow \{2\}$ ,  $\{2, 3\} \rightarrow \{1\}$ ,  $\{1\} \rightarrow \{2, 3\}$ ,  $\{2\} \rightarrow \{1, 3\}$  e  $\{3\} \rightarrow \{1, 2\}$ . Como o suporte de cada uma delas é idêntico ao de  $X$ , essas regras satisfazem o limite mínimo de suporte.

O cálculo da confiança de cada regra de associação não envolve nenhum percurso ou leitura adicional da base de dados transacional. Considere a regra  $\{1, 2\} \rightarrow \{3\}$ , a qual é gerada a partir do conjunto de itens frequente  $X = \{1, 2, 3\}$ . A confiança para essa regra é  $\sigma(\{1, 2, 3\})/\sigma(\{1, 2\})$ . Como  $\{1, 2, 3\}$  é frequente, a propriedade de monotonicidade do suporte garante que  $\{1, 2\}$  também seja frequente. Portanto, dado que a contagem de suporte para os dois conjuntos de itens já foi encontrada durante a geração dos conjuntos de itens frequentes, não há necessidade de percorrer toda a base de dados novamente.

#### Corte baseado na confiança

Diferentemente da métrica de suporte, a confiança não possui a propriedade de monotonicidade. Por exemplo, a confiança para  $X \rightarrow Y$  pode ser maior, menor ou igual à confiança de outra regra  $\tilde{X} \rightarrow \tilde{Y}$ , onde  $\tilde{X} \subseteq X$  e  $\tilde{Y} \subseteq Y$ .

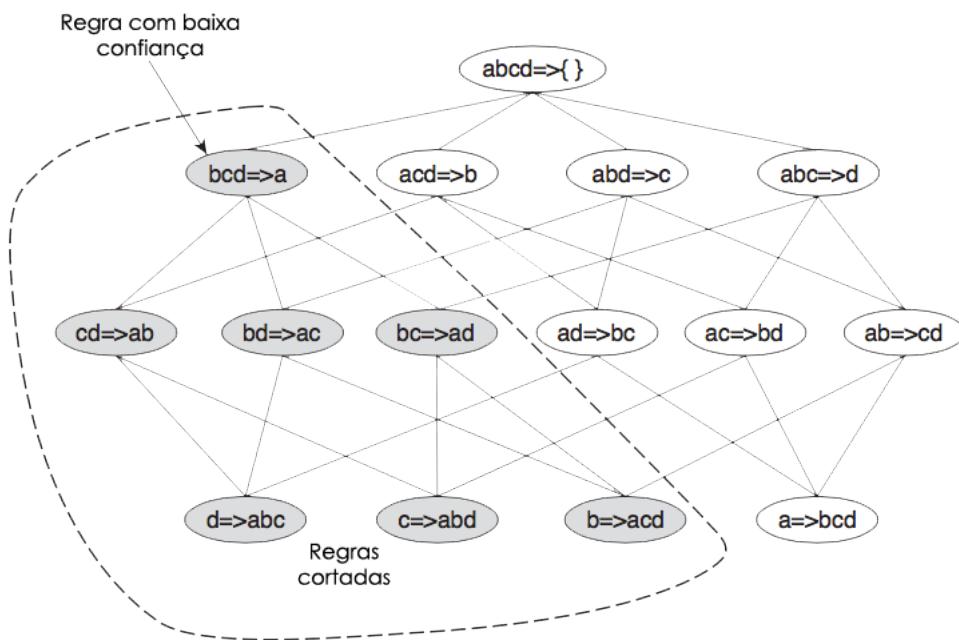
Considere as seguintes regras:  $X' \rightarrow Y \setminus X'$  e  $X \rightarrow Y \setminus X$ , onde  $X' \subset X$ . A confiança das regras é dada por  $\sigma(Y)/\sigma(X')$  e  $\sigma(Y)/\sigma(X)$ , respectivamente. Como  $X'$  é um subconjunto de  $X$ ,  $\sigma(X') \geq \sigma(X)$ . Portanto, a primeira regra não pode ter uma confiança maior que a última regra. A partir dessa definição, é possível elaborar o seguinte teorema:

**Teorema 2:** Se uma regra  $X \rightarrow Y \setminus X$  não satisfaz o limite mínimo de confiança, então qualquer regra  $X' \rightarrow Y \setminus X'$ , onde  $X'$  é um subconjunto de  $X$ , também não deve satisfazer o limite mínimo de confiança.

### Geração de regras no algoritmo *Apriori*

O algoritmo *Apriori* utiliza uma abordagem inteligente de níveis para gerar regras de associação, onde cada nível corresponde ao número de itens que pertencem ao consequente da regra. Inicialmente, todas as regras de alta confiança que possuem apenas um item em seu consequente são extraídas. Essas regras são então usadas para gerar novas regras candidatas. Por exemplo, se  $\{acd\} \rightarrow \{b\}$  e  $\{abd\} \rightarrow \{c\}$  são regras de alta confiança, então a regra candidata  $\{ad\} \rightarrow \{bc\}$  é gerada a partir da combinação dos consequentes das duas regras anteriores. A Figura 29 mostra a estrutura de grade para as regras de associação geradas a partir do conjunto de itens frequente  $\{a, b, c, d\}$ . Se qualquer nó nessa estrutura possuir uma baixa confiança, então, de acordo com o Teorema 2, o sub-grafo inteiro pertencente ao nó pode ser cortado imediatamente. Suponha que a confiança para  $\{bcd\} \rightarrow \{a\}$  seja baixa. Todas as regras contendo o item  $a$  em seu consequente, incluindo  $\{cd\} \rightarrow \{ab\}$ ,  $\{bd\} \rightarrow \{ac\}$ ,  $\{bc\} \rightarrow \{ad\}$  e  $\{d\} \rightarrow \{abc\}$ , serão descartadas.

Figura 29 – Corte de regras de associação baseado na métrica de confiança.



Fonte: (TAN; STEINBACH; KUMAR, 2005). Traduzida e adaptada pelo autor.

Outros pseudo-códigos para clarear a etapa de geração de regras são apresentados nos Algoritmos 6 e 7. A diferença neste procedimento em relação à geração de conjuntos de itens frequentes, é que não há necessidade de iterações adicionais ao longo da base de dados para efetuar o cálculo de confiança das regras candidatas. Logo, a confiança de cada regra é determinada utilizando a contagem de suporte computada durante a geração dos conjuntos de itens frequentes.

**Algoritmo 6** – GERAÇÃO DE REGRAS DO ALGORITMO *Apriori*

1. **Para** cada conjunto de  $k$  itens frequente  $f_k$ ,  $k \geq 2$  **faça**
2.      $H_1 = \{i \mid i \in f_k\}$ .
3.     **ap-geraregra**( $f_k$ ,  $H_1$ )

**Algoritmo 7** – PROCEDIMENTO **ap-geraregra**( $f_k$ ,  $H_m$ )

1.  $k = |f_k|$
2.  $m = |H_m|$
3. **Se**  $k > m + 1$  **então**
4.      $H_{m+1} = apriorigen(H_m)$
5.     **Para** cada  $h_{m+1} \in H_{m+1}$  **faça**
6.          $conf = \sigma(f_k)/\sigma(f_k - h_{m+1})$
7.         **Se**  $conf \geq minconf$  **então**
8.             **Gerar** a regra  $(f_k - h_{m+1}) \rightarrow h_{m+1}$
9.         **Se** **não**
10.             **Eliminar**  $h_{m+1}$  de  $H_{m+1}$
- 11.
12.     **ap-geraregra**( $f_k$ ,  $H_{m+1}$ )

O Algoritmo 6 é responsável por chamar a função geradora de regra para cada conjunto de  $k$  itens frequente  $f_k$  contido na base de dados, e o define como o antecedente de cada regra a ser criada. Enquanto isso, o procedimento ilustrado pelo Algoritmo 7 é responsável por chamar a função geradora de conjuntos de itens frequentes, *apriorigen*, e a partir de cada conjunto encontrado é então calculado sua confiança. Caso este valor esteja acima do valor mínimo de confiança definido, a regra será gerada para os dois conjuntos em questão, sendo o encontrado pela função *apriorigen* o consequente da regra. Caso este valor esteja abaixo do valor mínimo, o conjunto é eliminado e o algoritmo prossegue para a análise do próximo conjunto gerado pela função *apriorigen*.

# 3 Desenvolvimento

A aplicação de regras de associação fornece valiosas informações para a resolução dos problemas citados ao longo desse projeto. A grande versatilidade dessas regras, tanto no modo como são associadas como na forma em que são empregadas, nos auxilia a compreender e solucionar algumas situações mais complexas.

Muito desejada por empresas dos mais variados setores, essa técnica é capaz de recuperar diversas informações já armazenadas pelo usuário, ou seja, não há a necessidade da criação de uma nova base de dados para o problema, sendo apenas necessária a modelagem dos atuais dados existentes para que eles possam estar no mesmo formato de entrada compreendido pelos algoritmos utilizados.

Conforme proposto inicialmente, foi utilizada uma base de dados real composta por transações de diferentes produtos de uma loja de departamentos do estado do Rio de Janeiro. Também foram utilizadas bases sintéticas, ou seja, bases que possuem dados gerados apenas para fins científicos. Embora essas bases não possuam caráter real, elas também fornecem a mesma confiabilidade de uma base de dados real, pois os mesmos algoritmos e técnicas serão aplicados à ela, gerando resultados embasados pela mesma fundamentação matemática utilizada na outra base. A diferença consiste apenas no espaço de busca utilizado, já que as bases sintéticas possuirão dados e transações de caráter fictício.

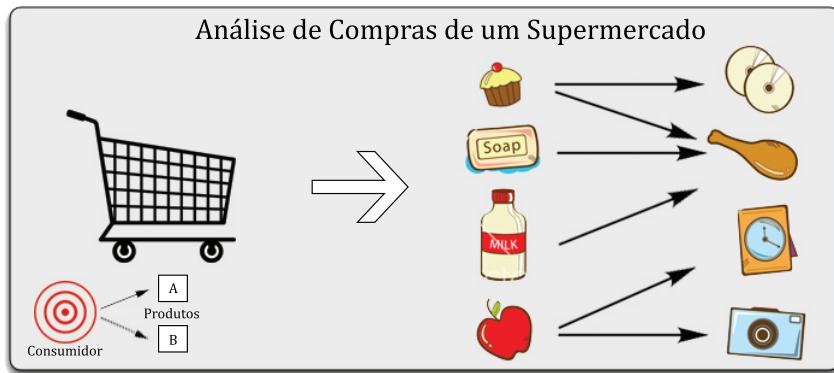
A seguir foram descritas com maiores detalhes as bases de dados utilizadas no projeto, incluindo a metodologia por trás dessas bases, sua origem, as informações as quais elas contém e como foram utilizadas dentro da resolução de nosso problema. Posteriormente, será discutida a implementação dos algoritmos *Apriori* ([BORGELT, 2003](#)) e *FP-Growth* ([BORGELT, 2005](#)) e sua execução dentro da esfera do projeto. E por fim, serão detalhados com tabelas, gráficos e análises estatísticas os resultados obtidos a partir da mineração das bases de dados utilizadas.

## 3.1 Método de Pesquisa

Esta seção descreve a metodologia empregada no presente trabalho de conclusão de curso. Conforme descrito na Seção [1.2.3](#), existem, basicamente, diversas abordagens para tentar solucionar alguns dos problemas da mineração de dados utilizando o modelo descritivo, podendo ser categorizadas conforme seguinte: (i) séries temporais, (ii) agrupamento, (iii) síntese, (iv) regras de associação e (v) descoberta de padrões e sequências. A abordagem utilizada no presente projeto de pesquisa enquadra-se dentro das regras de associação, onde através da análise das informações contidas no espaço do problema foram obtidas associações que promoveram o cruzamento de informações desconexas em formas mais comprehensíveis para

a geração de uma possível solução para o problema. Portanto, ao final do processo de criação das regras de associação, os resultados foram mensurados através de taxas de confiabilidade e suporte, proporcionando uma melhor tomada de decisão ao utilizar os mesmos.

Figura 30 – Análise de compras de um supermercado.



Fonte: Elaborada pelo autor.

A ideia consiste, basicamente, em utilizar informações estatísticas armazenadas em um banco de dados de um problema real, por exemplo, a utilização de informações históricas de compras e consumidores de um determinado supermercado, como demonstrado pela Figura 30. A partir destes dados foram empregados diversos algoritmos para a criação de regras de associação, ou seja, os algoritmos foram construídos para inferir e ordenar logicamente as informações desconexas existentes nesta base de dados, resultando em associações reais e plausíveis de serem entendidas e utilizadas para tentar solucionar o problema enfrentado.

## 3.2 Bases de Dados

### 3.2.1 Abordagem do problema real

O grande incentivo dado à análise de dados vem proporcionando diversos desejos nos mais variados perfis de usuários. Contudo, nem sempre é possível obter uma representação sintética desses dados, devendo ser coletados manualmente e posteriormente transformados para o meio digital.

Para se obter uma base de dados real é necessária a interpretação de dados também reais, por exemplo, oriundos das atividades de venda de uma loja de departamentos.

A metodologia proposta para a base de dados real origina a partir das tuplas armazenadas no banco de dados da empresa utilizada que por motivos contratuais não poderá ter seu nome divulgado. Entretanto, essa loja possui espaços físicos no estado do Rio de Janeiro e é responsável por efetuar vendas dos mais variados produtos no atacado, desde artigos alimentícios à até mesmo móveis e eletrodomésticos para o lar.

Como trata-se de uma base de dados real, a qual contém o histórico de todas as transações efetuadas pela loja, a tabela principal do banco de dados possui diversos atributos que registram características e informações sobre a venda de seus produtos. Não obstante, alguns desses atributos por não fornecerem a informação central que precisamos, isto é, o código identificador do produto e a transação a qual pertence, poderão ser descartados, exigindo uma reformulação da base para o nosso padrão de dados de entrada.

Tabela 4 – Exemplo de uma tupla da base de dados original.

Atributo	Exemplo
ID	7575
LOJA	C279
DATA_VENDA	2013-04-03
HORA	14
PERIODO	TARDE
NUM_ITEM	2
CODIGO_MERCADORIA	42281001
QNT_VENDIDA	1000
VALOR_TOTAL_BRUTO	249
TIPO_DESCONTO	NENHUM
VALOR_DESCONTO	0
NEGOCIO_COD	O101
NEGOCIO_DESCRICAO	TABLETES DE CHOCOLATE
CATEGORIA_CODIGO	O1
CATEGORIA_DESCRICAO	BOMBONIERE
MUNDO_COD	CONVENIENCIA
MUNDO_DESCRICAO	CONVENIENCIA
DESCRICAO_MERCADORIA	Barra Choc Baton Garoto 76g, Br/Cook

Fonte: Elaborada pelo autor.

A base original contém 2.194.306 registros de vendas de itens. Dentre todos esses registros, existem 14.647 diferentes itens que estão mapeados em 7 diferentes categorias. Contudo, como os registros são individuais para cada item, é possível que o número total de transações seja menor dado que podem existir transações que contenham mais de um item vendido. Portanto, apesar de apresentarmos na Tabela 4 a modelagem atual dessa base de dados, será necessária uma remodelagem de seu conteúdo a fim de assimilarmos cada item à sua transação, e assim reduzirmos o número de atributos empregados.

O modelo de banco de dados utilizado para o problema apresentado foi o modelo relacional, com a utilização da linguagem SQL para a modelagem dos dados. O gerenciador de banco de dados envolvido no problema foi o MySQL, o qual foi empregado a partir do uso de uma aplicação de alto nível para a sua manipulação, o Navicat MySQL.

A geração de regras de associação, de acordo com o modelo formal proposto na Seção 2.2.1, leva apenas em consideração os itens consumidos pelos usuários, ou seja, as

transações que contém apenas os produtos vendidos para aquela transação, não necessitando de informações como preços, quantidades, dentre outras. Portanto, a partir dessa base de dados original foi extraída uma determinada quantidade de registros e criadas as transações que serão analisadas pelos algoritmos utilizados. A Figura 31 ilustra a criação das sub-bases de acordo com as categorias existentes: Bricolagem, Casa, Conveniência, Eletro, Informática, Lazer, Você. Cada categoria possuirá sua sub-base, fornecendo uma maior compreensão sobre áreas mais específicas de nosso problema e não somente dele como um todo.

Figura 31 – Escopo das categorias utilizadas como sub-bases.



Fonte: Elaborada pelo autor.

A transação contém um campo identificador chamado TID, seguido pelos códigos dos itens que estão presentes nela. Por exemplo, imagine uma transação que contenha a venda de uma barra de chocolate (código 4228100) e um saco de amendoim (código 18776002). Após sua reformulação, o formato dos dados dessa transação segue o padrão estabelecido pela Tabela 5.

Tabela 5 – Exemplo de uma transação da base de dados remodelada.

<b>TID</b>	<b>Conjunto de Itens</b>
1	{4228100, 18776002}

Fonte: Elaborada pelo autor.

### 3.2.2 Abordagem do problema sintético

O capitalismo é responsável por forçar uma grande competitividade entre seus correntes, minimizando a quantidade de informações concretas disponíveis e eliminando a possibilidade de se obter dados reais para a análise e criação de regras de associação. A falta de informações sólidas, a dificuldade em modelar sistemas ideais ou até mesmo problemas que enfrentamos diariamente motivaram a criação de bases de dados artificiais, ou seja, representações de dados criadas sinteticamente com o intuito de preencher as lacunas deixadas pelas empresas portadoras dessas informações.

A existência dessas bases de dados projetadas especificamente para determinado propósito nos auxiliam na compreensão da teoria estudada e também na execução da parte prática da mesma. Portanto, serão descritas nessa seção todo o problema sintético criado e toda a estrutura necessária para a sua execução.

Sistemas de recomendação vêm sendo utilizados com o intuito de prever o desejo do consumidor e auxiliar nos processos de tomada de decisão das empresas. Nossa caso de uso baseia-se em uma empresa de telecomunicações que efetua a venda de pacotes para canais de TV e, recentemente, vem sofrendo com um declínio em sua quantidade de clientes. A fim de resolver esse problema, foram realizadas duas pesquisas: uma com 1.000 indivíduos a fim de saber quais são seus 50 canais de TV mais assistidos dentro da grade de programação da empresa (base de dados sintética 1k), e outra com 10.000 indivíduos contendo os seus 60 canais mais assistidos (base de dados sintética 10k). A partir desses dados, é possível minerá-los e criar algumas regras que ajudarão no ajuste dos atuais pacotes e na melhor disposição da grade de vendas da empresa. Por exemplo, após o emprego da metodologia proposta, será possível inferir que  $x\%$  clientes da empresa que assistem ao canal de esportes SporTV (pacote 2) também assistem ao canal de esportes ESPN (pacote 6), mostrando um possível interesse desses indivíduos em receber uma oferta que combine a utilização de ambos os canais.

As informações já foram tratadas e o banco de dados reduzido ao mínimo necessário para a aplicação de nossa metodologia. As Tabelas 6, 7 e 8 descrevem com maiores detalhes as estruturas e tabelas utilizadas na base criada.

Tabela 6 – Exemplo de uma tupla da tabela ‘usuarios’ da base sintética.

Atributo	Exemplo
ID	64
NOME	Larissa Lopes
SEXO	F
IDADE	27
PROFISSAO	MUSICO
QNT_FILHOS	0

Fonte: Elaborada pelo autor.

A tabela ‘usuarios’ é constituída por um campo identificador ID, o nome do indivíduo, seu sexo (M ou F), sua idade (intervalo de 18 a 85 anos), sua profissão (14 classes) e sua quantidade de filhos (0 a 5 filhos). As profissões estão enquadradas nas seguintes categorias: ADVOGADO, APOSENTADO, ARQUITETO, ARTISTA, EMPRESARIO, ENGENHEIRO, ESCRITOR, ESTUDANTE, MEDICO, MUSICO, PROFESSOR, PSICOLOGO, SECRETARIO, VENDEDOR.

Tabela 7 – Exemplo de uma tupla da tabela ‘canais’ da base sintética.

Atributo	Exemplo
ID	109
NOME	HBO Plus
GENERO	FILME
PACOTE	4

Fonte: Elaborada pelo autor.

A tabela ‘canais’ é constituída por um campo identificador ID, o nome do canal, seu gênero (14 classes) e o pacote a qual pertence (1 ao 7). Os gêneros estão enquadrados nas seguintes categorias: ADULTO, EDUCACAO, ESPORTE, FILME, GERAL, INFATIL, INFORMACAO, MUNDO, MUSICA, POLITICA, RELIGIAO, SERIE, VARIADO, VENDA. E a tabela ‘canais\_usuarios’ é responsável por mapear as preferências de canais de todos os usuários da base de dados. Cada tupla é composta por um campo identificador ID, o ID do usuário e o ID do canal que foi escolhido em uma de suas preferências.

Tabela 8 – Exemplo de uma tupla da tabela ‘canais\_usuarios’ da base sintética.

Atributo	Exemplo
ID	1
USUARIO_ID	4
CANAL_ID	101

Fonte: Elaborada pelo autor.

A transação final conterá um campo identificador chamado TID, seguida pelos códigos dos canais que estão presentes nela. Por exemplo, imagine uma transação de um indivíduo que assista mais aos canais BIS (código 31), Fox Life (código 27), Futura (código 54) e TNT (código 96). Portanto, o formato dos dados dessa transação segue o padrão estabelecido pela Tabela 9.

Tabela 9 – Exemplo de uma transação da base de dados sintética remodelada.

TID	Conjunto de Canais
1	{31, 27, 54, 96}

Fonte: Elaborada pelo autor.

A distribuição dos dados foi feita de forma aleatória a partir de um algoritmo de distribuição probabilística desenvolvido para o próprio projeto<sup>1</sup>. Os canais dos pacotes foram distribuídos com o intuito de aproximar os dados gerados de um problema real, ou seja, canais de pacotes mais baratos e de maior acesso à população aparecerão mais ao longo da base de dados enquanto que canais de pacotes mais caros (*pay-per-view*) aparecerão em uma menor quantidade. A Tabela 10 exemplifica a distribuição utilizada, e a Tabela 11 é responsável por mapear os identificadores dos canais contidos em cada pacote.

Tabela 10 – Distribuição da porcentagem de pacotes empregada na geração dos dados.

Pacote	Quantidade
1	50%
2	25%
3	12.5%
4	3.125%
5	3.125%
6	3.125%
7	3.125%

Fonte: Elaborada pelo autor.

Tabela 11 – Distribuição de canais por pacote.

Pacote	Listagem de Canais
1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 17, 18, 19, 20, 21, 22, 23, 24, 25, 39, 40, 41, 45, 46, 49, 50 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 117
2	26, 27, 28, 29, 30, 31, 47, 48, 65, 66, 67, 68, 69, 70, 71, 72 73, 74, 79, 80, 81, 87, 88, 89, 90, 91, 92, 96, 97, 100, 118, 119 120, 121, 122, 123, 124, 131, 132
3	42, 43, 44, 93, 94, 95, 98, 99, 125 126, 127, 128, 129, 130, 133, 134
4	35, 101, 102, 103, 104, 105, 106, 107, 108 109, 110, 111, 112, 113, 114, 115, 116
5	32, 33, 34, 36, 37, 38
6	82, 83, 84, 85, 86
7	75, 76, 77, 78

Fonte: Elaborada pelo autor.

<sup>1</sup> [https://github.com/gugarosa/regras\\_associacao\\_tcc/blob/master/bases\\_dados/gerador\\_base\\_sintetica.c](https://github.com/gugarosa/regras_associacao_tcc/blob/master/bases_dados/gerador_base_sintetica.c)

### 3.3 Algoritmos Utilizados

#### 3.3.1 *Apriori*

O algoritmo *Apriori*, por ser o estudo base da associação de regras e por sua fácil aplicação, ainda é muito utilizado em problemáticas reais de menor escala e também em situações de cunho científico. A metodologia proposta por esse algoritmo executa uma busca exploratória incremental ao longo dos possíveis níveis existentes de conjuntos de itens, em outras palavras, ele percorre uma altura por vez partindo de conjuntos de um item até conjuntos de tamanho  $k_{max}$ . Entretanto, esse algoritmo ainda realiza expansões desnecessárias quando empregado em espaços de busca extensos, ocasionando em uma redução de sua eficiência computacional.

A implementação utilizada neste projeto advém do trabalho proposto e criado por Christian Borgelt ([BORGELT, 2003](#)), o qual foi comprado pela empresa multinacional IBM e empregado no conhecido software de análise e predição de dados denominado *Statistical Package for the Social Sciences* (SPSS). Porém, a versão utilizada é a original criada por Borgelt, disponível em seu site<sup>2</sup>, dado que toda sua implementação está feita na linguagem C, sendo possível efetuar as alterações necessárias para que a biblioteca possa ser empregada dentro da esfera deste projeto.

Os experimentos realizados neste trabalho permitiram um maior entendimento sobre o seu funcionamento, mostrando as principais situações em que o algoritmo é realmente eficiente e em quais casos ele deixa a desejar. A partir dos resultados apresentados, também é possível inferir o maior problema relacionado à esse trabalho, apresentado e discutido diversas vezes ao longo desta literatura, o qual é composto pela complexidade exponencial do espaço de busca, ou seja, a dificuldade em analisar com eficiência grandes quantidades de dados.

#### 3.3.2 *FP-growth*

Apresentado e embasado pela Seção 2.2.4, o algoritmo *FP-growth* utiliza uma metodologia contrária ao da técnica *Apriori*, onde a base de dados é totalmente codificada para uma nova estrutura de dados denominada árvore FP, a qual foi utilizada como base da execução do algoritmo. Assim como o seu concorrente *Apriori*, existirão casos em que esse algoritmo será mais eficiente, por exemplo bases mais extensas com uma variação menor de itens ao longo das transações, e também situações em que deixará a desejar, por exemplo bases que contenham diferentes itens para cada transação, dado que serão necessários muitos acessos aos nós da estrutura. Para este algoritmo, a implementação utilizada também advém de outro trabalho proposto e criado por Christian Borgelt ([BORGELT, 2005](#)). Novamente a versão utilizada é a original criada por Borgelt, disponível em seu site<sup>3</sup>, e também implementada na linguagem C.

<sup>2</sup> <http://www.borgelt.net/apriori.html>

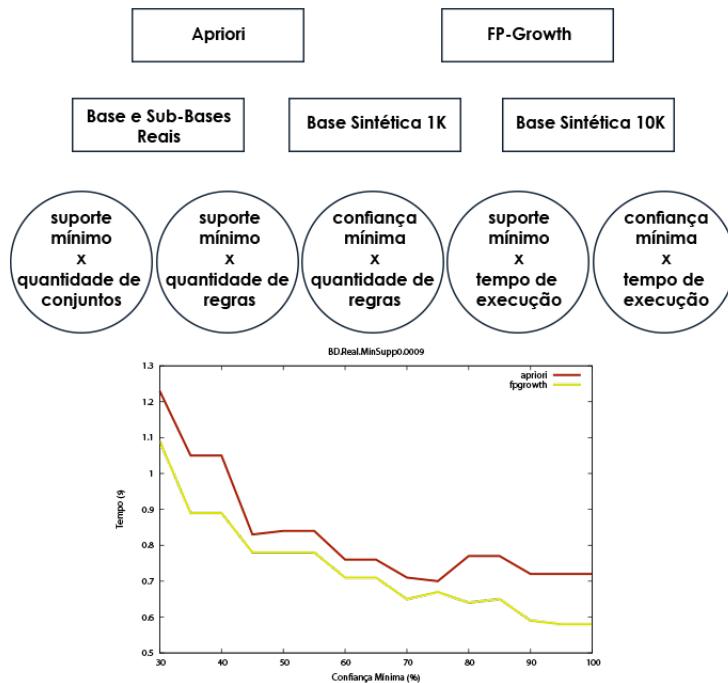
<sup>3</sup> <http://www.borgelt.net/fpgrowth.html>

Portanto, a seguir, serão apresentados diversos experimentos realizados nas bases de dados reais e sintéticas propostas por este trabalho, capazes de fornecer uma melhor comparação entre os algoritmos utilizados.

### 3.4 Experimentos

A possibilidade da utilização de uma base de dados real e outras sintéticas fornece à este trabalho uma maior robustez, dado que muitas vezes em problemáticas computacionais os algoritmos tendem a operar de diferentes formas em diferentes bases de dados, ou seja, cada problema tem a sua especificidade. Entretanto, ainda é possível observar características e resultados em comum ao longo de suas operações.

Figura 32 – Detalhamento dos experimentos realizados.



Fonte: Elaborada pelo autor.

Os experimentos foram realizados em ambos os algoritmos *Apriori* e *FP-Growth*. Cada conjunto de experimentos foi aplicado às diferentes bases de dados apresentadas: real, sub-bases reais, sintética com 1.000 indivíduos (1K) e sintética com 10.000 indivíduos (10K). Cada conjunto de experimentos é composto por análises de duas dimensões (2D) para uma posterior representação em gráficos XY. Nestas análises estão inclusas: suporte mínimo x quantidade de conjuntos encontrados, suporte mínimo x quantidade de regras encontradas, confiança mínima x quantidade de regras encontradas e por fim os conjuntos de experimentos que nos permitirão uma melhor comparação entre os algoritmos, suporte mínimo x tempo de execução e confiança mínima x tempo de execução. Todos os experimentos foram realizados utilizando o sistema

operacional Ubuntu Linux, em uma máquina com processador Intel Core i7 (8 núcleos) e 8GB de memória RAM. A Figura 32 fornece o detalhamento dos experimentos propostos.

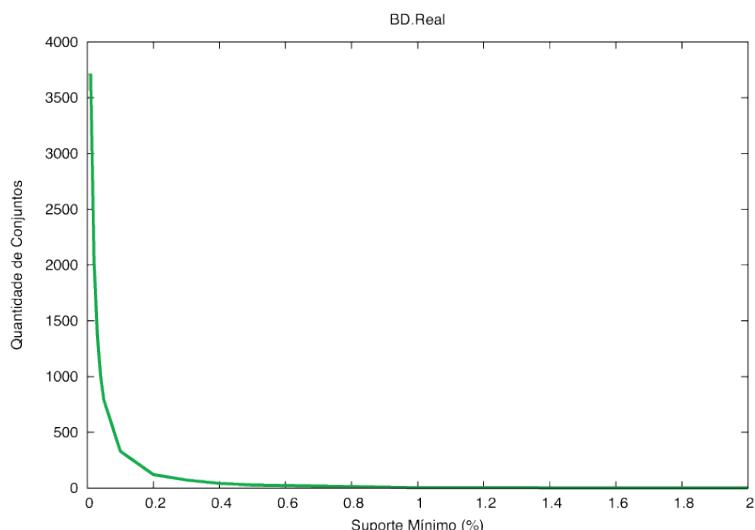
Como a metodologia proposta depende basicamente de apenas duas variáveis, suporte e confiança, foram empregados vários valores de intervalos numéricos a fim de se obter resultados mais robustos e confiáveis ao problema. Cada base de dados possuirá intervalos específicos de operação, mas sempre levando em consideração o mesmo ambiente de execução e parametragem para ambos os algoritmos, fato necessário para analisar e comparar suas complexidades. Todos os experimentos apresentados, exceto os contidos na Seção 3.5.4, são válidos para ambos os algoritmos estudados, dado que suas gerações de conjuntos e regras foram idênticas, apenas diferindo no modo como foram encontrados e no tempo de execução. Finalmente, também serão apresentadas por extenso algumas regras geradas e conhecimentos obtidos a partir da análise das bases de dados propostas, mostrando o efetivo poder da mineração de dados e o porquê dela ser tão desejada pelas empresas nos dias de hoje.

## 3.5 Resultados

### 3.5.1 Base real

Como apresentado pela Seção 3.2.1, foi utilizada uma base de dados advinda de uma verdadeira loja de varejo com o intuito de testar a metodologia empregada e verificar se os resultados obtidos condizem com a fundamentação proposta para essa abordagem. Inicialmente, foi efetuada uma contagem do número de conjuntos frequentes obtidos em diferentes intervalos numéricos utilizados para a variável de suporte.

Figura 33 – Gráfico do suporte mínimo x quantidade de conjuntos para a base de dados real.



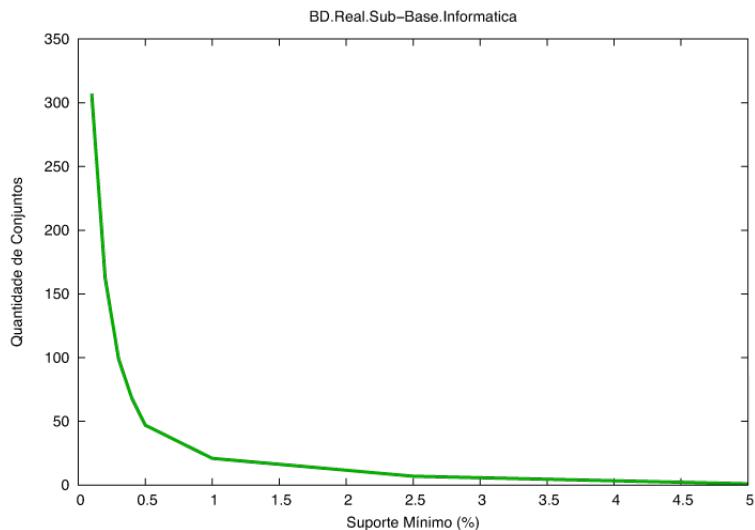
Fonte: Elaborada pelo autor.

Com base nesse experimento, foi possível observar a complexidade exponencial do

espaço de busca e a importante necessidade de não somente gerar conjuntos frequentes, mas sim efetuar cortes e otimizar o processo para a geração de conjuntos eficientes. A quantidade de conjuntos cresce exponencialmente cada vez que são utilizados valores menores de suporte, até determinado ponto onde os próprios algoritmos empregados não conseguem efetuar suas análises devido ao grande número de conjuntos que poderão ser gerados. A Figura 33 demonstra um gráfico que melhor exemplifica esse processo. Nesse experimento, foram utilizados valores de suporte entre 0.01 e 2.00, valor máximo o qual foi capaz de gerar apenas um conjunto frequente.

As Figuras 34 e 35 apresentam as contagens de conjuntos frequentes para duas das sub-bases obtidas a partir da divisão da base de dados real, sendo elas a sub-base informática e a sub-base conveniência, respectivamente.

Figura 34 – Gráfico do suporte mínimo x quantidade de conjuntos para a sub-base de dados informática.

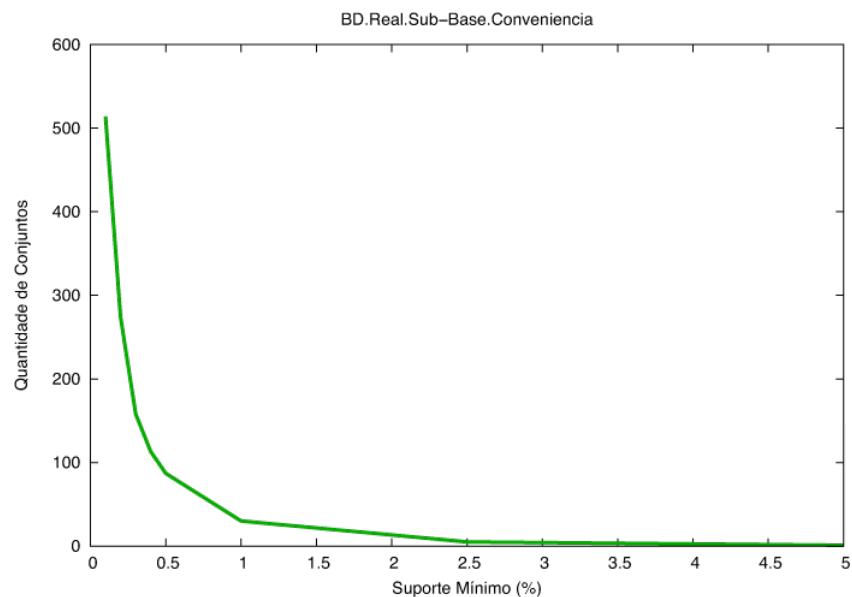


Fonte: Elaborada pelo autor.

Voltando ao problema da base de dados real, também foram efetuados experimentos para identificar a quantidade de regras geradas a partir da utilização de diferentes valores para a variável suporte. Como dito anteriormente, é de grande interesse não apenas gerar quaisquer regras, mas sim regras que serão eficazes e que poderão ser utilizadas para auxiliar no processo de tomada de decisão da empresa.

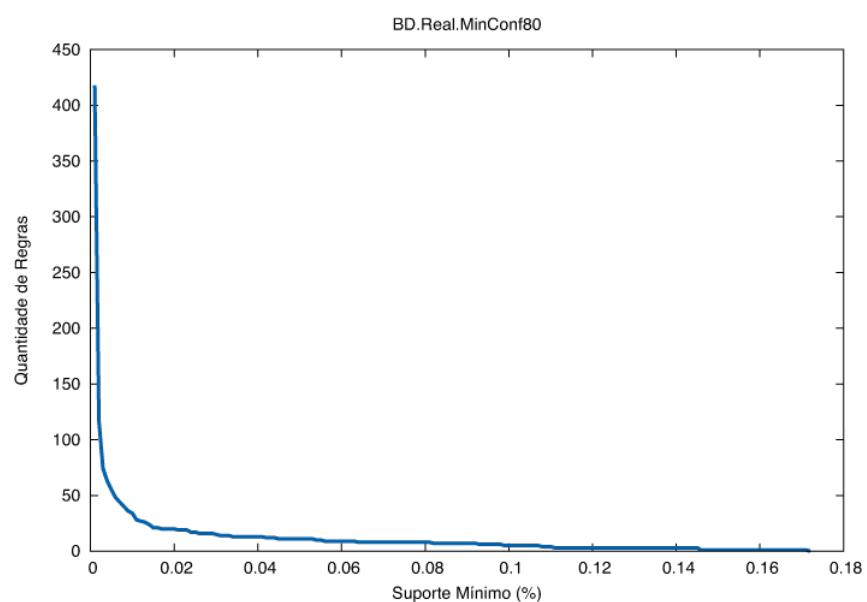
As Figuras 36, 37 e 38 são responsáveis por ilustrar esse procedimento para diferentes valores mínimos de confiança empregados. Como pode-se notar ao passar dos gráficos, cada vez que a variável de confiança recebe um valor maior, menos regras serão geradas, ou seja, ao passo que o suporte funciona como uma variável de corte no espaço de busca, a confiança também serve para efetuar cortes, gerando assim regras mais eficazes e mais prováveis de serem úteis para a resolução do problema.

Figura 35 – Gráfico do suporte mínimo x quantidade de conjuntos para a sub-base de dados conveniência.



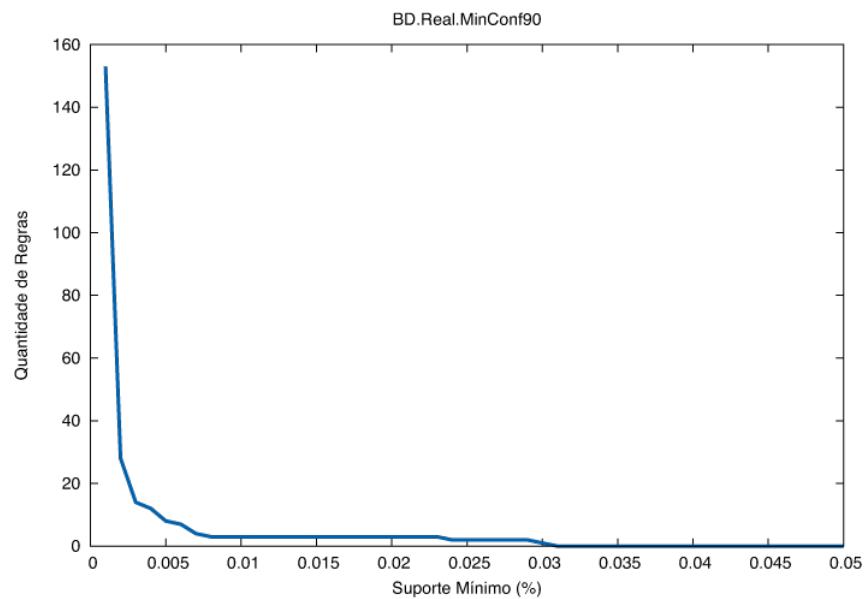
Fonte: Elaborada pelo autor.

Figura 36 – Gráfico do suporte mínimo x quantidade de regras para a base de dados real (confiança mínima 80%).



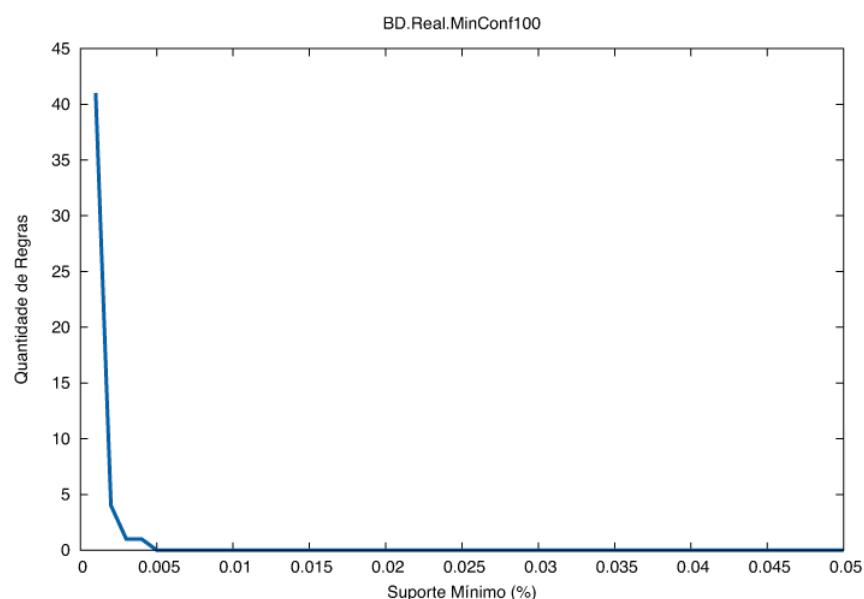
Fonte: Elaborada pelo autor.

Figura 37 – Gráfico do suporte mínimo x quantidade de regras para a base de dados real (confiança mínima 90%).



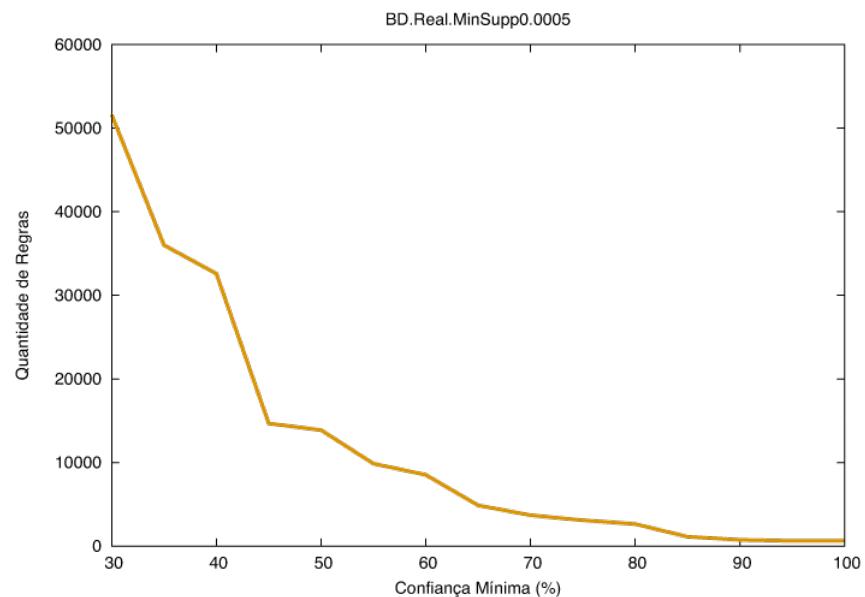
Fonte: Elaborada pelo autor.

Figura 38 – Gráfico do suporte mínimo x quantidade de regras para a base de dados real (confiança mínima 100%).



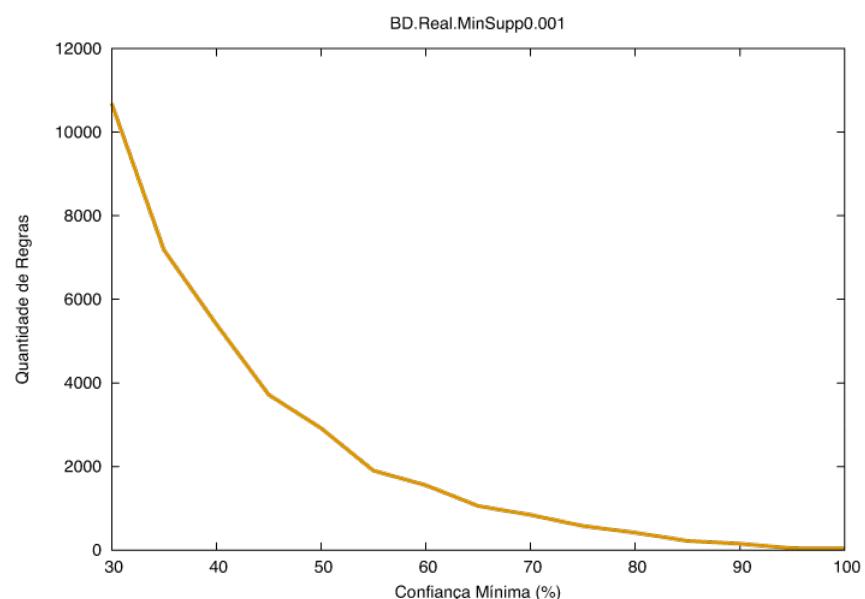
Fonte: Elaborada pelo autor.

Figura 39 – Gráfico da confiança mínima x quantidade de regras para a base de dados real (suporte mínimo 0.0005%).



Fonte: Elaborada pelo autor.

Figura 40 – Gráfico da confiança mínima x quantidade de regras para a base de dados real (suporte mínimo 0.001%).



Fonte: Elaborada pelo autor.

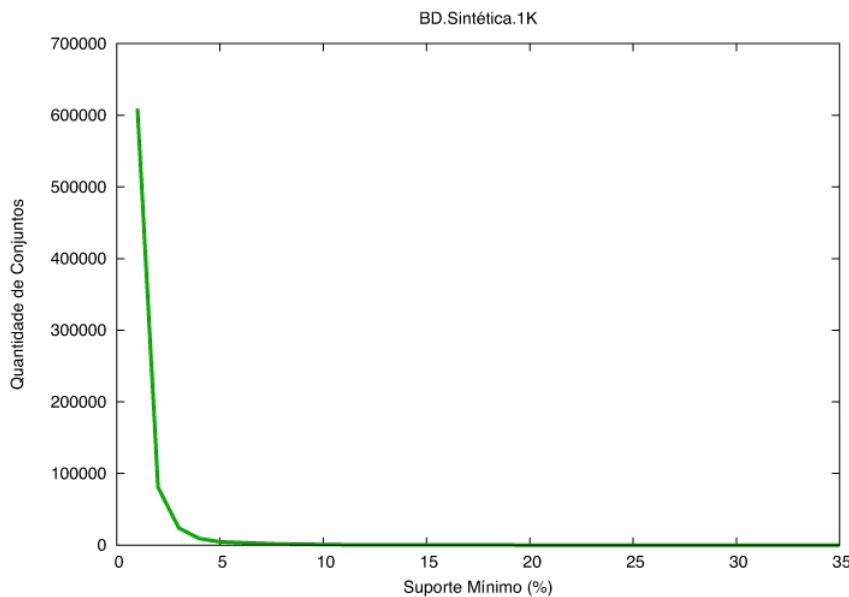
Entretanto, também é de grande interesse analisar o que acontece quando fixamos algum determinado valor para a variável de suporte e consideramos diferentes valores mínimos para a confiança. As Figuras 39 e 40 apresentam os experimentos obtidos para essa abordagem. Como pode-se notar, os valores fixados para a variável de suporte são em torno de cem vezes menores do que os utilizados no experimento anterior. Isso deve-se ao fato de que foi necessária a utilização de valores menores para aumentar a quantidade de regras produzidas, proporcionando uma melhor observação no decaimento da quantidade de regras geradas.

### 3.5.2 Base sintética 1K

A falta de dados reais para serem analisados motivou a criação de uma base de dados sintética, fornecendo mais informações acerca do problema apresentado e trazendo resultados mais robustos. A criação dessa base está embasada na Seção 3.2.2, sendo que os mesmos experimentos efetuados para a base de dados real também foram desenvolvidos aqui.

Novamente, foi feito a princípio uma contagem da quantidade de conjuntos frequentes encontrados pelos algoritmos *Apriori* e *FP-Growth* para diferentes valores de suporte, comprovando o crescimento exponencial da base à medida que são utilizados valores menores para a variável de corte (suporte).

Figura 41 – Gráfico do suporte mínimo x quantidade de conjuntos para a base de dados sintética 1K.

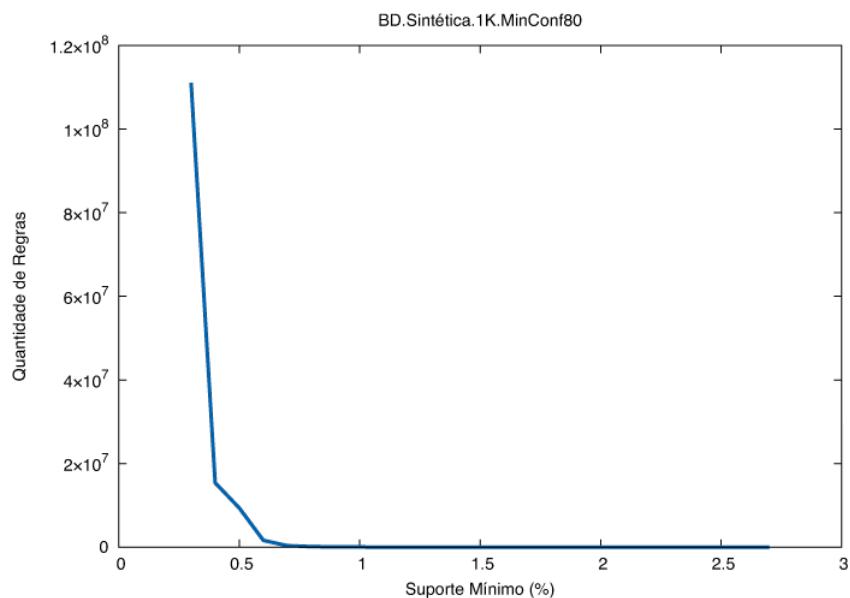


Fonte: Elaborada pelo autor.

A partir disso, também foram efetuados experimentos para comprovar o crescimento exponencial da geração de regras, mostrando que a geração é bem simples de ser efetuada, porém, não tão simples quando se quer gerar regras eficientes e que possam ser empregadas para a correta resolução do problema. A quantidade de regras geradas demonstrada pelas Figuras 42, 43 e 44 ilustra o atual problema da mineração de dados, o qual se encaixa na

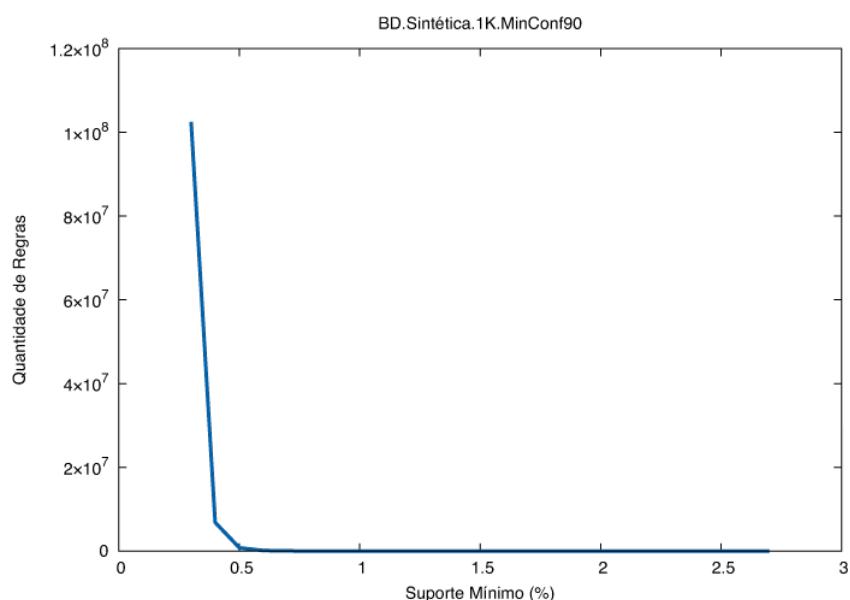
quantidade exponencial de informações que são criadas a partir da mineração dos dados da base.

Figura 42 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 1K (confiança mínima 80%).



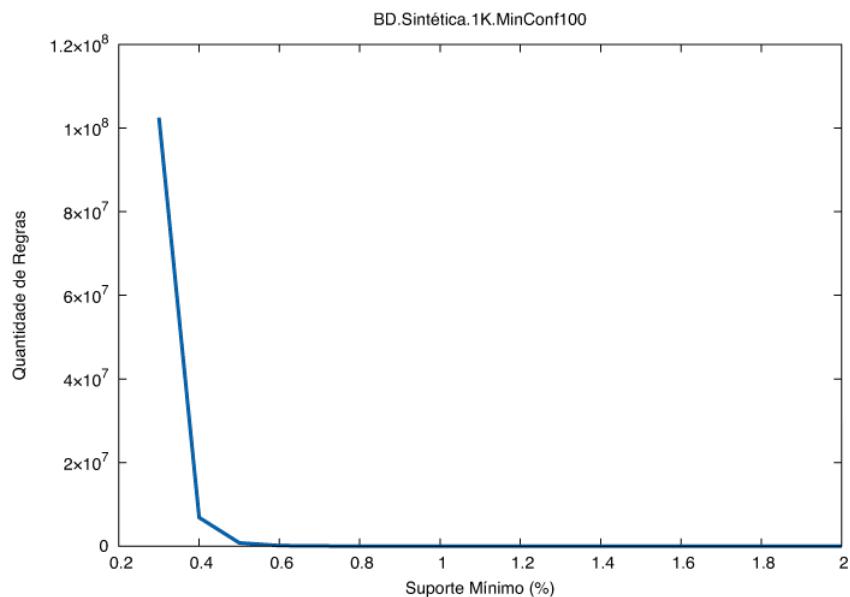
Fonte: Elaborada pelo autor.

Figura 43 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 1K (confiança mínima 90%).



Fonte: Elaborada pelo autor.

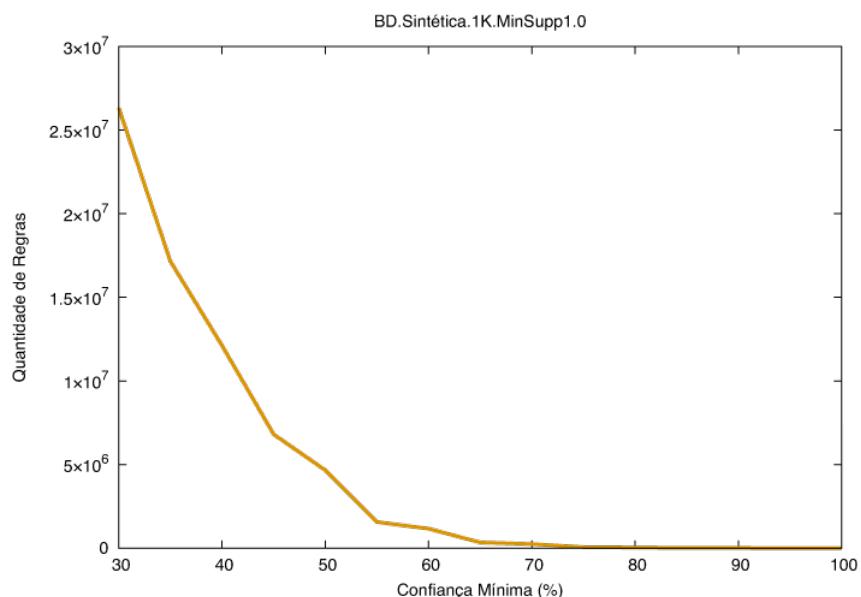
Figura 44 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 1K (confiança mínima 100%).



Fonte: Elaborada pelo autor.

E, por fim, também é de grande interesse avaliar os resultados quando fixamos um valor para a variável de suporte e consideramos diferentes valores mínimos para a confiança. A Figura 45 ilustra esse procedimento para um valor mínimo de suporte de 1%.

Figura 45 – Gráfico da confiança mínima x quantidade de regras para a base de dados sintética 1K (suporte mínimo 1%).



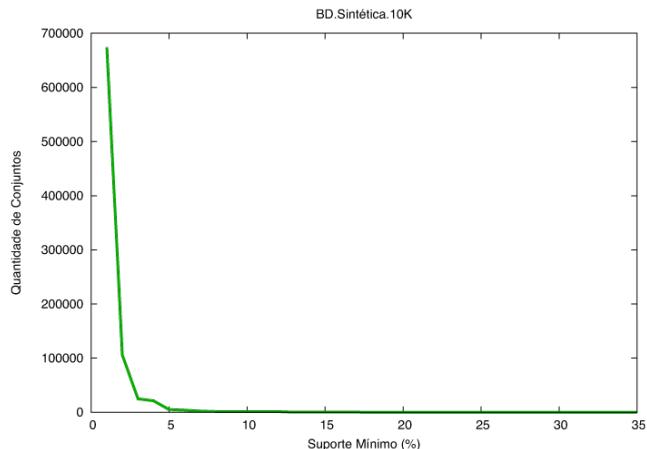
Fonte: Elaborada pelo autor.

### 3.5.3 Base sintética 10K

A criação de uma base sintética mais robusta e com um maior nível de informações também foi necessária para esse trabalho, pois há uma necessidade de avaliar casos sintéticos que se aproximem dos casos reais, ou seja, é preciso avaliar uma base de dados sintética que possua uma grande quantidade de informações, assim como uma base de dados de uma empresa real.

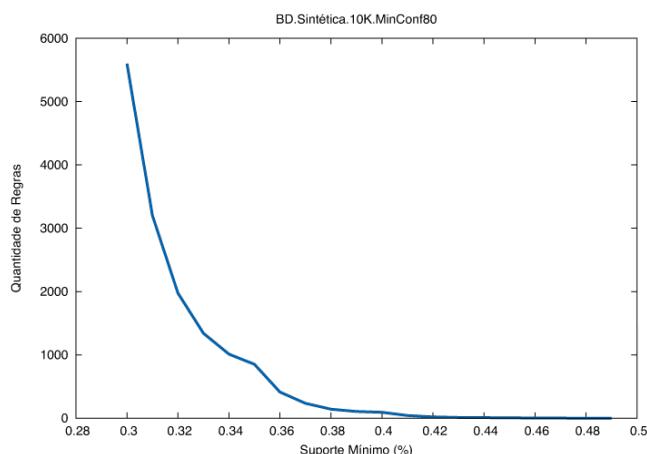
Os mesmos experimentos conduzidos nas seções anteriores também foram aplicados para essa base de dados. A diferença consiste nos intervalos numéricos utilizados para as variáveis de suporte e confiança, dado que cada base possui seus próprios melhores valores de intervalos. As Figuras 46, 47, 48 e 49 ilustram esses experimentos.

Figura 46 – Gráfico do suporte mínimo x quantidade de conjuntos para a base de dados sintética 10K.



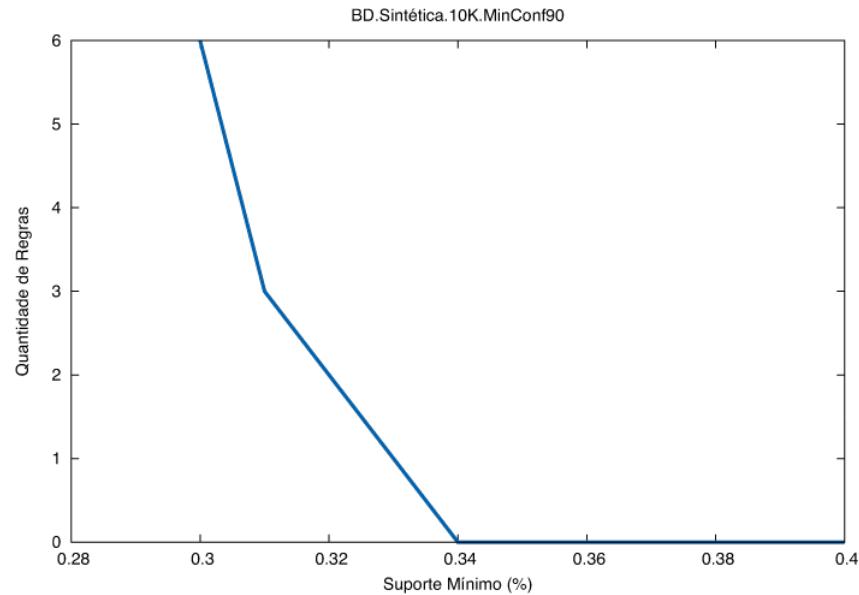
Fonte: Elaborada pelo autor.

Figura 47 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 10K (confiança mínima 80%).



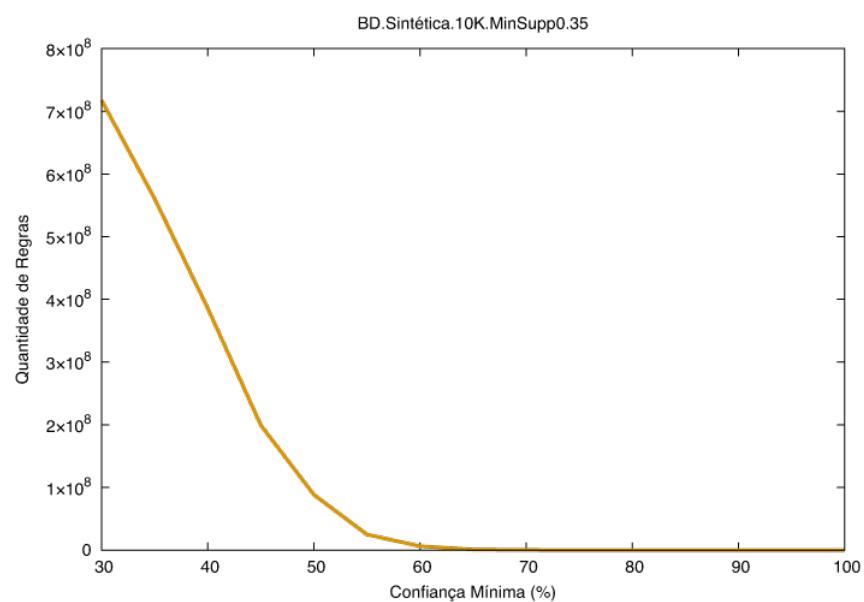
Fonte: Elaborada pelo autor.

Figura 48 – Gráfico do suporte mínimo x quantidade de regras para a base de dados sintética 10K (confiança mínima 90%).



Fonte: Elaborada pelo autor.

Figura 49 – Gráfico da confiança mínima x quantidade de regras para a base de dados sintética 10K (suporte mínimo 0.35%).

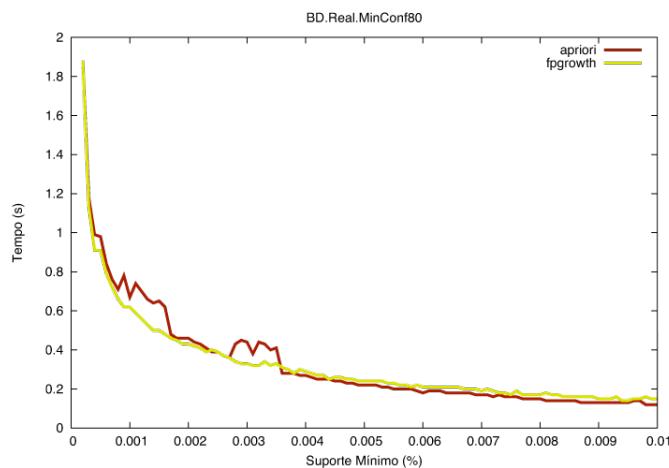


Fonte: Elaborada pelo autor.

### 3.5.4 Tempo de execução

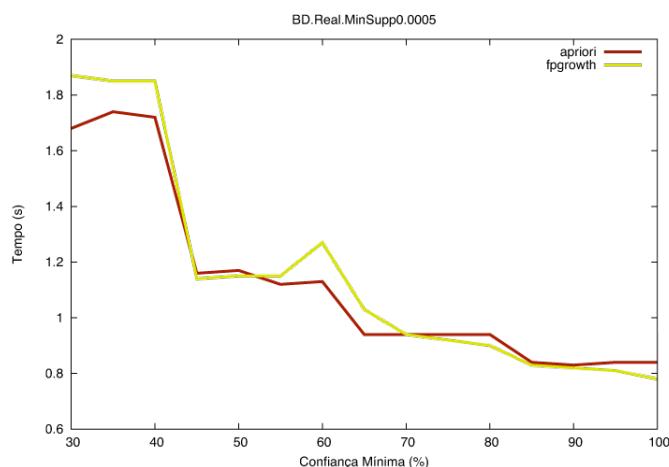
Nesta seção, serão descritos os experimentos relativos à comparação entre os algoritmos *Apriori* e *FP-Growth*. Como mencionado na Seção 3.3, existem casos em que o *Apriori* melhor executa sua metodologia, porém também existem casos em que o *FP-Growth* é superior ao seu concorrente. Diversas contagens de tempo foram efetuadas em todas as bases de dados utilizadas, mas somente as principais serão ilustradas pelos gráficos a seguir, mostrando um comparativo no tempo de execução de ambos algoritmos para os mais variados experimentos propostos pela Seção 3.4. As Figuras 50, 51, 52, 53, 54 e 55 ilustram esses experimentos.

Figura 50 – Gráfico do suporte mínimo x tempo de execução para a base de dados real (confiança mínima 80%).



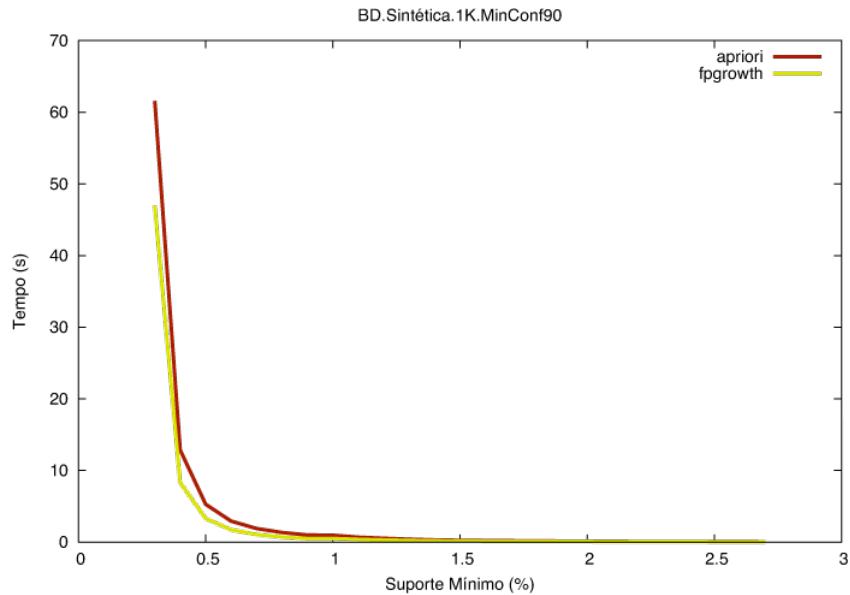
Fonte: Elaborada pelo autor.

Figura 51 – Gráfico da confiança mínima x tempo de execução para a base de dados real (suporte mínimo 0.0005%).



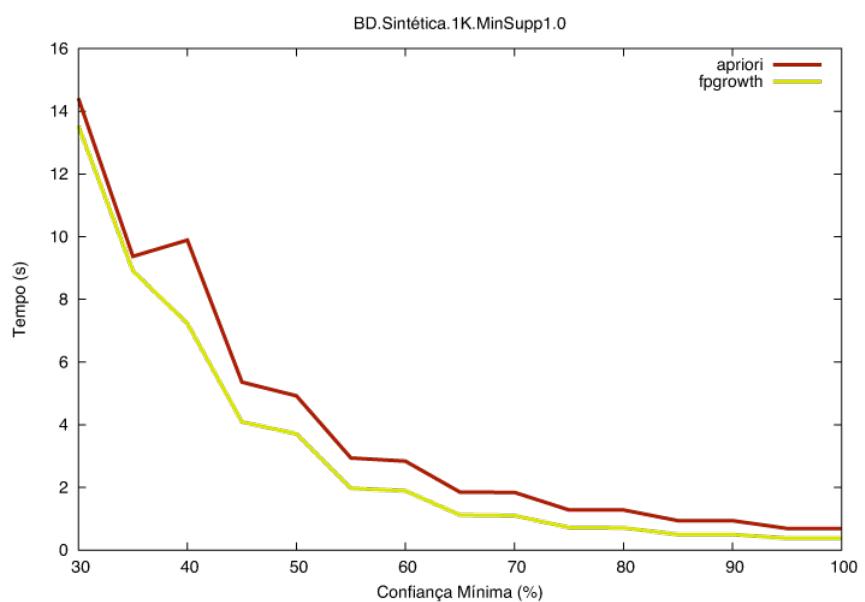
Fonte: Elaborada pelo autor.

Figura 52 – Gráfico do suporte mínimo x tempo de execução para a base de dados sintética 1K (confiança mínima 90%).



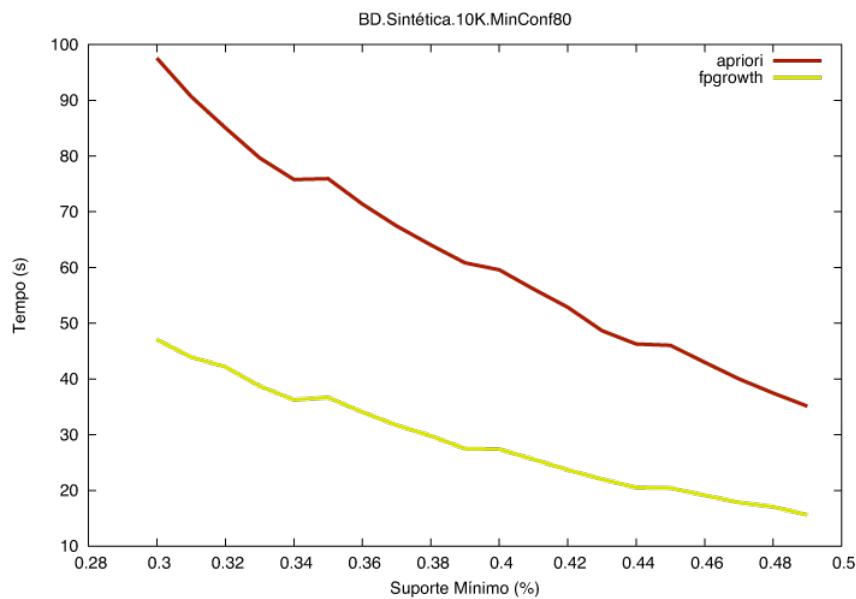
Fonte: Elaborada pelo autor.

Figura 53 – Gráfico da confiança mínima x tempo de execução para a base de dados sintética 1K (suporte mínimo 1%).



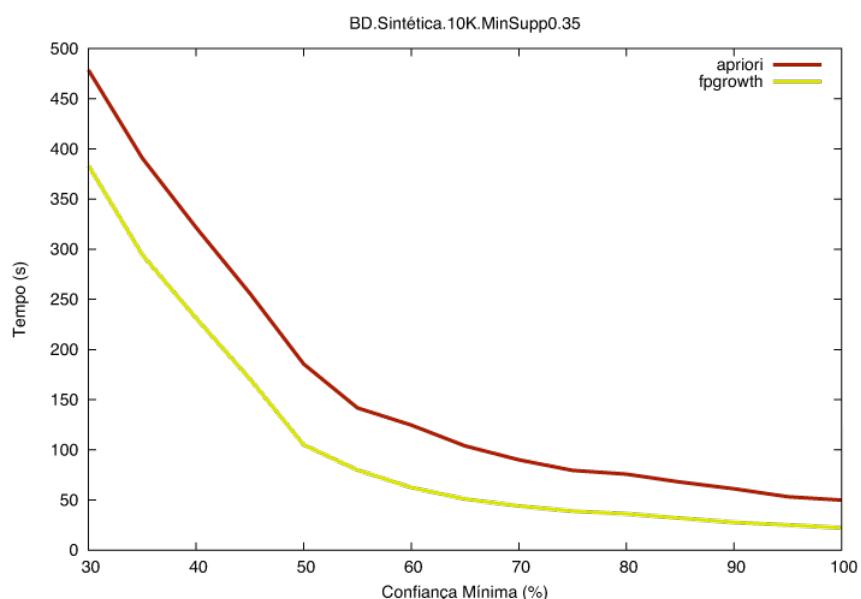
Fonte: Elaborada pelo autor.

Figura 54 – Gráfico do suporte mínimo x tempo de execução para a base de dados sintética 10K (confiança mínima 80%).



Fonte: Elaborada pelo autor.

Figura 55 – Gráfico da confiança mínima x tempo de execução para a base de dados sintética 10K (suporte mínimo 0.35%).



Fonte: Elaborada pelo autor.

Os gráficos acima mostram que os algoritmos *Apriori* e *FP-Growth* possuem tempos de execução similares para a maioria dos casos, diferindo em poucos segundos. Entretanto, ao avaliar uma base de dados mais extensa, como por exemplo a sintética 10K, o algoritmo *FP-Growth* necessita de um tempo de execução bastante inferior em relação ao seu concorrente, principalmente devido ao seu mecanismo exploratório de busca dos conjuntos frequentes, dado que o mesmo não precisa avaliar todos os possíveis conjuntos de tamanho  $k = 1$  até  $k_{max}$ .

### 3.6 Regras Geradas

Além dos gráficos e tabelas produzidos a partir dos experimentos efetuados, outro importante resultado é a geração das próprias regras. Nesta seção serão descritas algumas das regras geradas a partir da metodologia empregada para todas as bases de dados. Entretanto, apenas serão consideradas e descritas as regras de maior importância para a resolução do problema, isto é, as regras mais eficazes dentro de suas categorias e que, por conseguinte, possuem valores maiores de suporte e confiança.

Tabela 12 – Regras com maiores valores de suporte e confiança para a base de dados real.

Regra	Suporte (%)	Confiança (%)
TRosto Senegal Azul, TRosto Senegal Vermelha, TBanho Senegal Azul → TBanho Senegal Vermelha	0.0022	100
Samsung Chat 222 Pt Vivo → Chip Pre Vivo	0.0044	100
Alcatel OT255 2Ch Pt Vivo → Chip Pre Vivo	0.0238	97.85
Modem USB 3G Huawei Tim → Chip Pre Infinity 21 Tim	0.0308	92.72
Samsung E1182 Duos Basic Pr Vivo → Chip Pre Vivo	0.0295	92.39
Cel Alcatel Desb OT217 2Ch Pt → Chip ODA Pre Claro	0.1451	86.72
Cel Freecel Desb Free One 2Ch Pt → Chip ODA Pre Claro	0.1718	81.25

Fonte: Elaborada pelo autor.

Pode-se observar na Tabela 12 que as regras geradas com maiores valores de suporte e confiança possuem itens inerentes à categoria de Informática e, principalmente, relativos à telefonia celular. A partir dessa análise, é possível recomendar a nossa empresa que eles efetuam um melhor planejamento de sua área de informática, dado que a maior quantidade de produtos vendidos origina-se nessa área, e também uma ótima proposição seria efetuar parcerias com empresas de telefonia celular, oferecendo espaço à elas dentro de suas lojas e aumentando a quantidade de clientes que procuram por esses serviços, em especial a Claro, dado que esta empresa aparece com itens de maior frequência (maior suporte) ao longo das regras obtidas.

Também há a possibilidade de gerar regras com mais itens em seu antecedente ou consequente, porém quanto maior a complexidade da regra, menor será sua taxa de suporte e confiança. A criação de regras com altos valores de suporte e confiança são de extrema eficiência para a abordagem direta do problema. Contudo, também são interessantes as regras

com as menores taxas, principalmente de confiança, pois com elas poderão ser encontrados determinados padrões que possam existir na loja ou até mesmo que ainda não existam, auxiliando a inferir novos conhecimentos.

Tabela 13 – Regras com maiores valores de suporte e confiança para a base de dados sintética 1K.

Regra	Suporte (%)	Confiança (%)
Nick Jr. (2), Disney Channel (2), Ideal TV (1) → FOX (2)	1.1	100
Sony (2), Curta! (1), Canal Rural (1), Ideal TV (1) → NBR (1)	1.1	100
Glitz* (2), TV Brasil (1), TV Justiça (1) → Off (1)	1.9	94.74
FX (2), Rede Vida (1), TV Câmara (1) → NBR (1)	2.1	90.48
Space (2), GNT (1), SHOPBUY (1) → Multishow (1)	2.2	81.82
Warner Channel (2), Off (1), TV Justiça (1) → MTV (2)	2.3	82.61

Fonte: Elaborada pelo autor.

A Tabela 13 mostra algumas das regras com maiores valores de suporte e confiança para a base de dados sintética 1K. Pode-se observar que a nossa base está composta por muitos estudantes universitários, pessoas que têm preferência por canais políticos e variados, e indivíduos que possuem filhos, fato que pode ser comprovado ao analisarmos a tabela ‘usuarios’ da base, identificando que 12% dela está composta por estudantes e que 71% da base possui ao menos um filho. A partir dessas regras, é possível efetuar algumas recomendações para a nossa empresa fictícia, onde seria de grande interesse aliar canais infantis e séries (pacote 2) aos pacotes mais comuns oferecidos, por exemplo, ao pacote 1, que é a base de operação da empresa. Mesmo sendo o pacote mais simples e barato de todos, muitos optam por não somente assistí-lo, dado que os melhores canais estão contidos nos outros pacotes oferecidos.

Tabela 14 – Regras com maiores valores de suporte e confiança para a base de dados sintética 10K.

Regra	Suporte (%)	Confiança (%)
TV Aparecida (1), Curta! (1), Band (1), Rede do Bem (1), Rede Mundial (1), Shoptime (1) → Canal Universitário (1)	0.3	90
Play TV (1), Mix TV (1), TV Brasil (1), RIT TV (1), TV Senado (1), Shoptime (1) → Canal Legislativo (1)	0.3	90
Rede 21 (1), Rede do Bem (1), TV Justiça (1), Record (1), RIT TV (1), Canal Legislativo (1) → TV Aberta (1)	0.33	81.82
Viva (1), NBR (1), Rede 21 (1), RDBTV (1), Record (1), Ideal TV(1) → TV Escola (1)	0.38	86.84
Discovery Turbo (3), Off (1), Band News (1), Rede do Bem (1), Leomax Shop (1) → TV Câmara (1)	0.41	85.37
Viva (1), TV Canção Nova (1), CNT (1), Multishow (1), Canal Universitário (1), Leomax Shop (1) → Globo (1)	0.45	80

Fonte: Elaborada pelo autor.

Finalmente, a Tabela 14 ilustra algumas das regras criadas a partir da utilização da maior base sintética, com 10,000 indivíduos. Embora algumas regras possam ser estranhas

ou não fizerem sentido, devemos lembrar que a distribuição de canais para cada usuário foi feita utilizando uma distribuição probabilística, de acordo com a Tabela 10. Portanto, as regras correspondem corretamente ao algoritmo gerador da base de dados, onde a maior proporção dos canais assistidos eram relativos ao pacote 1 e, por consequinte, apareceram com uma maior incidência nessa base de dados, possuindo os maiores valores de suporte. Isso demonstra a fragilidade da geração de regras, onde não é somente interessante avaliar seus níveis de suporte e confiança, mas também estabelecer outros limites ou restrições a fim de melhorar o processo de geração de regras e torná-las mais eficazes para a resolução do problema.

A base de dados sintética 10K foi desenvolvida justamente para exemplificar esse problema, que é contemplado pelo problema inicial do projeto, ou seja, a mineração de regras eficientes em bases de dados extensas. A partir das análises desenvolvidas em todos os experimentos para todas as bases de dados empregadas, foi possível concluir todo o estudo embasado na fundamentação teórica do projeto, demonstrando o passo-a-passo de como é efetuada a criação e associação de regras utilizando a mineração de dados, exemplificando os pontos fortes e fracos da metodologia utilizada.

## 4 Conclusão

O estudo na área da mineração de dados engloba diferentes conhecimentos dentro da própria computação, tornando-se uma área bastante completa e responsável por fornecer diversas ferramentas para a resolução dos mais variados problemas encontrados atualmente. A evolução tecnológica está proporcionando a possibilidade de efetuar novos estudos e aprofundar ainda mais em assuntos os quais ainda não podiam ser estudados, talvez devido à dificuldade de resolução do problema ou simplesmente por não haver um meio possível de simulá-lo e testá-lo. O crescimento do poder computacional fornece cada vez mais a capacidade de efetuar cálculos mais rápidos e assim efetuar procedimentos mais complexos, como, por exemplo, a mineração de dados.

A necessidade de técnicas mais robustas e resultados mais confiáveis são fatores que fomentam o estudo nessa área, dado que ela está diretamente ligada ao mercado, o qual está fortemente influenciado pelo capitalismo selvagem que nos circunda. Entretanto, o aumento do poder computacional não traz somente benefícios à pesquisa nessa área, mas ele também é responsável por acarretar um aumento na quantidade de dados que estão sendo analisados. A chamada "Internet das Coisas" e a alta interconectividade entre os dispositivos atuais influencia na quantidade do fluxo de dados que trafega na rede, contribuindo para um aumento exponencial no armazenamento dos dados, fato que está diretamente ligado à técnica de mineração de dados.

A dificuldade em analisar eficientemente grandes quantidades de dados, conceito tido por *Big Data*, foi um dos principais problemas avaliados por este trabalho. É de grande valia a criação e associação de regras eficazes que correspondem corretamente ao problema em questão e também são capazes de auxiliar no processo de tomada de decisão. Como abordado pelo trabalho, a associação de regras é um passo bastante simples se considerarmos sua metodologia, entretanto, a criação de regras eficientes não é uma tarefa fácil, envolvendo alguns impasses e paradigmas relacionados à sua execução.

A fundamentação teórica que embasa esse projeto traz consigo algumas abordagens e ferramentas para minimizar a quantidade de dados analisados e efetuar cortes eficazes ao longo da base, diminuindo a quantidade de regras geradas e maximizando a sua eficiência na resolução do problema. Os experimentos desenvolvidos por esse trabalho auxiliam na visualização do que está realmente acontecendo por trás da execução da metodologia estudada, e comprovam o crescimento exponencial da complexidade computacional ao se analisar grandes quantidades de dados, fator que dificulta a tarefa de criação e associação de regras.

As diferentes bases de dados empregadas, real e sintéticas, também são responsáveis por fornecer as diferentes problemáticas que podem ser confrontadas por uma mineração de

dados. Bases menores, bases maiores, problemas fictícios, problemas reais, toda informação diferente é de suma importância para a análise do método, dado que atualmente os dados podem ser encontrados nos mais diversos formatos.

Portanto, este projeto contempla uma coletânea de estudos e técnicas inerentes à Ciência da Computação na área de mineração de dados e regras de associação, ao passo que juntas contribuem para a formação de uma abordagem mais complexa e capaz de abordar alguns dos atuais desafios enfrentados pelo mercado. O material apresentado por este projeto poderá servir de apoio à criação de uma nova disciplina optativa, mais voltada para a parte prática da computação e do segmento do mercado. A utilização da mineração de dados não fornece somente conhecimentos de alto nível para seus usufruintes, mas também auxilia no processo de formação do aluno que a estuda, pois sempre haverá a necessidade de se empregar e combinar diversos conhecimentos matemáticos e computacionais; sendo assim, uma interessante área de pesquisa a ser seguida no futuro. Outros importantes trabalhos futuros consistem na utilização de novas técnicas de otimização para a geração de regras de associação, isto é, a utilização de algoritmos meta-heurísticos e inspirados pela natureza (i.e. otimização por enxame de partículas, busca harmônica, algoritmo do morcego, dentre outros) para a otimização dos parâmetros empregados nesta técnica, contribuindo para uma geração de regras mais eficientes e funcionais.

# Referências

- AGRAWAL, R. et al. An interval classifier for database mining applications. In: *Proceedings of the 18th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992. (VLDB '92), p. 560–573. ISBN 1-55860-151-1. Citado na página [24](#).
- AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Database mining: a performance perspective. *Knowledge and Data Engineering, IEEE Transactions on*, v. 5, n. 6, p. 914–925, Dec 1993a. Citado na página [17](#).
- AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 22, n. 2, p. 207–216, jun. 1993b. Citado 7 vezes nas páginas [17](#), [22](#), [23](#), [25](#), [55](#), [56](#) e [58](#).
- AGRAWAL, R. et al. Advances in knowledge discovery and data mining. In: FAYYAD, U. M. et al. (Ed.). [S.I.]: American Association for Artificial Intelligence, 1996. cap. Fast Discovery of Association Rules, p. 307–328. ISBN 0-262-56097-6. Citado na página [23](#).
- AGRAWAL, R.; PSAILA, G. Active data mining. In: *The First International Conference on Knowledge Discovery and Data Mining*. [S.I.]: AAAI Press, 1995. p. 3–8. Citado 2 vezes nas páginas [17](#) e [22](#).
- AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. (VLDB '94), p. 487–499. Citado 4 vezes nas páginas [20](#), [22](#), [55](#) e [61](#).
- ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, n. 15, p. 2787 – 2805, 2010. ISSN 1389-1286. Citado na página [21](#).
- BAYARDO JR., R. J.; AGRAWAL, R. Mining the most interesting rules. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 1999. (KDD '99), p. 145–154. ISBN 1-58113-143-7. Citado na página [62](#).
- BORGELT, C. Efficient implementations of apriori and eclat. In: *Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL)*. CEUR Workshop Proceedings 90. [S.I.: s.n.], 2003. p. 90. Citado 3 vezes nas páginas [55](#), [65](#) e [72](#).
- BORGELT, C. An implementation of the fp-growth algorithm. In: *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*. New York, NY, USA: ACM, 2005. (OSDM '05), p. 1–5. Citado 2 vezes nas páginas [65](#) e [72](#).
- BURDICK, D.; CALIMLIM, M.; GEHRKE, J. Mafia: a maximal frequent itemset algorithm for transactional databases. In: *Data Engineering, 2001. Proceedings. 17th International Conference on*. [S.I.: s.n.], 2001. p. 443–452. Citado na página [54](#).

- COFFMAN JR., E. G.; EVE, J. File structures using hashing functions. *Commun. ACM*, ACM, New York, NY, USA, v. 13, n. 7, p. 427–432, jul. 1970. Citado na página 41.
- HAN, J. et al. Dmql: A data mining query language for relational databases. In: . [S.I.: s.n.], 1996. p. 27–33. Citado na página 24.
- HAN, J.; KAMBER, M. *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. ISBN 1-55860-489-8. Citado na página 18.
- HAN, J.; PEI, J.; YIN, Y. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 29, n. 2, p. 1–12, maio 2000. Citado na página 46.
- HOLSHEIMER, M. et al. *A Perspective on Databases and Data Mining*. Amsterdam, The Netherlands, The Netherlands, 1995. Citado na página 24.
- LIN, D.-I.; KEDEM, Z. M. Pincer search: A new algorithm for discovering the maximum frequent set. In: *Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology*. London, UK, UK: Springer-Verlag, 1998. (EDBT '98), p. 105–119. Citado na página 52.
- LIU, B.; HSU, W.; MA, Y. Integrating classification and association rule mining. In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA),: AAAI Press, 1998. p. 80–86. Citado na página 22.
- MANNILA, H.; TOIVONEN, H.; VERKAMO, I. Efficient algorithms for discovering association rules. In: . [S.I.]: AAAI Press, 1994. p. 181–192. Citado na página 26.
- MOORE, G. E. Cramming More Components onto Integrated Circuits. *Electronics*, IEEE, v. 38, n. 8, p. 114–117, abr. 1965. ISSN 0018-9219. Citado na página 19.
- PARK, J. S.; CHEN, M.-S.; YU, P. S. An effective hash-based algorithm for mining association rules. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 24, n. 2, p. 175–186, maio 1995. Citado na página 41.
- PASQUIER, N. et al. Discovering frequent closed itemsets for association rules. In: *Proceedings of the 7th International Conference on Database Theory*. London, UK, UK: Springer-Verlag, 1999. (ICDT '99), p. 398–416. ISBN 3-540-65452-6. Citado na página 42.
- PIATESKI, G.; FRAWLEY, W. *Knowledge Discovery in Databases*. Cambridge, MA, USA: MIT Press, 1991. ISBN 0262660709. Citado na página 23.
- SAVASERE, A.; OMIECINSKI, E.; NAVATHE, S. B. An efficient algorithm for mining association rules in large databases. In: *Proceedings of the 21th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. (VLDB '95), p. 432–444. ISBN 1-55860-379-4. Citado na página 26.
- SRIKANT, R.; VU, Q.; AGRAWAL, R. Mining association rules with item constraints. In: . [S.I.]: AAAI Press, 1997. p. 67–73. Citado na página 28.
- STONEBRAKER, M. et al. Dbms research at a crossroads: The vienna update. In: *VLDB '93 Proceedings of the 19th International Conference on Very Large Data Bases*. [S.I.]: Morgan Kaufmann Publishers Inc., 1993. p. 688–692. Citado na página 16.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367. Citado 19 vezes nas páginas [23](#), [26](#), [30](#), [31](#), [32](#), [33](#), [34](#), [38](#), [39](#), [40](#), [41](#), [43](#), [47](#), [48](#), [50](#), [52](#), [53](#), [54](#) e [63](#).

TOIVONEN, H. Sampling large databases for association rules. In: *Proceedings of the 22th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. (VLDB '96), p. 134–145. ISBN 1-55860-382-4. Citado na página [26](#).

WANG, K.; HE, Y.; HAN, J. Mining frequent itemsets using support constraints. In: *Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. (VLDB '00), p. 43–52. ISBN 1-55860-715-3. Citado na página [36](#).

ZAKI, M. et al. Evaluation of sampling for data mining of association rules. In: *Research Issues in Data Engineering, 1997. Proceedings. Seventh International Workshop on*. [S.l.: s.n.], 1997. p. 42–50. Citado na página [26](#).

ZAKI, M. J. Generating non-redundant association rules. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2000. (KDD '00), p. 34–43. Citado na página [44](#).