Descoberta de Conhecimento em Bancos de Dados

Marco Eugênio Madeira Di Beneditto

17 de junho de 2003

Relatório da matéria MAC5701 - Tópicos em Ciência da Computação

Orientador: Prof. Dr. Leliane Nunes de Barros

1. Introdução

A *Descoberta de Conhecimento em Bancos de Dados* é um processo não trivial de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis incluídos nos dados [Fayyad et al., 1996]. Para compreender esta definição é necessário examinar outras definições, a saber:

- *dados* correspondem ao conjunto de fatos em um Base de dados, por exemplo, os valores dos campos de um registro;
- *padrão* é uma expressão, em alguma linguagem, que descreve um subconjunto dos dados ou um modelo aplicável a este subconjunto;
- identificar ou extrair um padrão significa descobrir uma descrição de alto nível de um conjunto de dados.
- por *processo*, subentende-se um conjunto de etapas.
- *não trivial* implica, não apenas o cálculo de quantidades predefinidas, mas o envolvimento de alguma busca ou inferência.
- a *validade* decorre da necessidade de haver alguma medida de certeza em relação aos padrões descobertos.
- padrões que sejam *novos* são mais desejados, principalmente pelo usuário;
- padrões *potencialmente úteis* conduzem a algum benefício ao usuário ou a um sistema, e finalmente,
- padrões devem ser *compreensíveis*, imediatamente ou após algum processamento por um especialista no domínio dos dados.

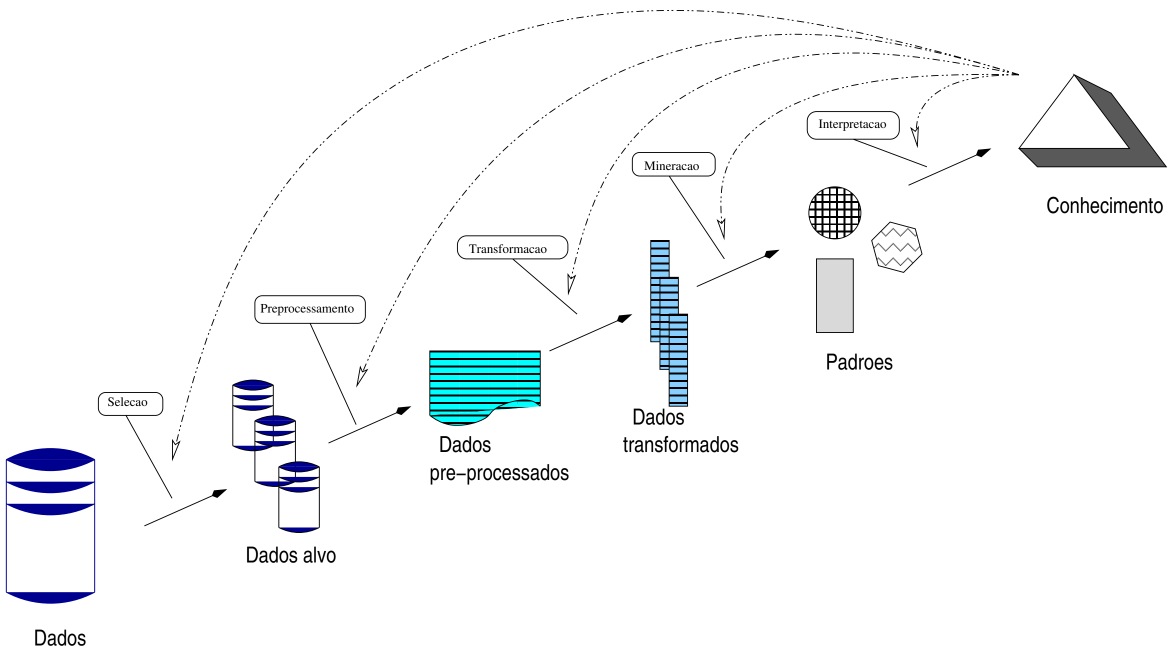

Figura 1: Etapas do processo de Descoberta de Conhecimento

Os objetivos de um sistema de descoberta de conhecimento podem ser: (i) a predição, onde os padrões são descobertos com o intuito de prever o comportamento futuro de novos dados, ou (ii) a descrição, onde o padrão visa apresentar ao usuário uma visão concisa dos dados.

Na figura 1, o processo de descoberta de conhecimento é dividido em cinco etapas. As linhas pontilhas mostram que o conhecimento, prévio ou o próprio conhecimento descoberto, pode fornecer subsídios a todas as etapas do processo. Nas etapas de *seleção*,*preprocessamento* e *transformação* dos dados, o conhecimento pode reduzir o número de exemplos necessários, agrupando-os em conjuntos que representam um conceito mais elevado, bem como diminuir a dimensão do problema identificando variáveis irrelevantes ao domínio. Assim, na mineração de dados, o conhecimento prévio, além de ajudar na redução do espaço de busca por um padrão, permite obter conhecimento num nível de abstração mais elevado, compacto e muitas vezes mais interessante, como é desejado.

Um exemplo de conhecimento prévio pode ser uma *hierarquia de conceitos*, ou seja, uma estrutura em que conceitos de mais alto-nível de abstração (generalizações de conceitos de mais baixo nível) são organizados hierarquicamente. A maioria dos estudos em mineração de dados somente procura por padrões no nível conceitual representado no banco de dados que, nem sempre, é o nível de conhecimento que se deseja ou mesmo o nível que possui o maior grau de interesse. Dados com grande regularidade, isto é, com grande ocorrência num banco de dados, usualmente não aparecem em níveis conceituais baixos pois, neste caso, os padrões tendem a ser dispersos e com pequeno suporte. Como exemplo, têm-se os registros das compras efetuadas num supermercado onde apenas os códigos de barra são registrados, e dificilmente descobrem-se os padrões que associam os itens entre si numa mesma compra, como pão integral e leite desnatado ao invés do pão marca X e leite marca Y. Porém, dados num nível conceitual mais elevado podem levar a padrões que já são de conhecimento comum, não possuindo algo de novo.

Estas observações motivaram a proposta de descobrir conhecimento em múltiplos níveis conceituais, aumentando a diversidade dos padrões descobertos e permitindo ao usuário focar em pontos que lhe sejam mais interessantes [Taylor, 1999, Fu, 1996, Han et al., 1992].

Neste trabalho foi implementado um sistema para descoberta de regras de classificação empregando conhecimento prévio, representado através de hierarquias de conceitos sobre os atributos de uma base de dados.

2. Fundamentos

2.1. Padrões descobertos

Um padrão a ser descoberto pode ser representado por uma expressão, em alguma linguagem, descrevendo um subconjunto dos dados ou um modelo aplicável a este subconjunto. Dentre os padrões que podem ser descobertos temos as regras de classificação, caracterização, discriminação, regras de associação, generalização e agrupamentos. Neste trabalho, será tratado apenas a descoberta de regras de classificação nos processos de mineração em múltiplos níveis.

Definimos uma classificação por uma função de aprendizado que mapeia dados de entrada em um número finito de classes pré-definidas. A etapa de mineração de dados busca determinar esta função. Cada tupla consiste em um conjunto de atributos e, dentre eles, há um atributo ou mesmo um conjunto de atributos que determinam a classe à qual pertence a tupla. Uma maneira de expressar uma classificação é através de regras da forma: **se** A **então** C_n, onde A representa o conjunto de pares atributo-valor $\langle A_i, v_j \rangle$ tal que A_i é o i-ésimo atributo de uma tabela no banco de dados e v_j é um dos j-valores dentre os existentes no domínio de A_i, C_n é a n-ésima classe e C é o conjunto das classes C_n. Note que os conjuntos C e A são disjuntos.

2.2. Hierarquia de conceitos

Hierarquias de conceitos representam um tipo de conhecimento prévio que pode ser usado para a descoberta de conhecimento em múltiplos níveis.

Para que o conhecimento descoberto esteja num nível superior ao encontrado na base de dados, os conceitos e dados nesta base podem estar organizados em diferentes níveis de abstração. Tal organização pode ser definida por meio de uma hierarquia de conceitos especificada por um especialista do domínio de aplicação, ou do banco de dados. Uma hierarquia de conceitos define relações de generalização e especialização para conceitos em um atributo ou um conjunto de atributos. Embora uma hierarquia de conceitos possa ser fornecida por um especialista de domínio, esta também pode ser construída ou refinada dinamicamente [Fu, 1996]. Uma hierarquia de conceitos pode ser estabelecida a partir de um banco de dados das seguintes maneiras, onde $\preceq$ é uma relação de ordem parcial:

Hierarquia entre atributos Implicitamente armazenada no banco de dados, definindo níveis de generalidade entre os atributos como, por exemplo, o endereço residencial de um indivíduo representado pelos seguintes atributos que possuem uma ordem parcial, ou seja,
número da casa $\preceq$ *rua* $\preceq$ *bairro* $\preceq$ *cidade* $\preceq$ *estado*

Hierarquia entre valores de atributos Definida por um determinado conjunto de valores de um atributo, como mostra a hierarquia da figura 2 para o atributo *cor*. Um conceito de mais alto nível define a tonalidade *escura* para um conjunto de cores,
$\{$*canela, chocolate*$\}$ $\preceq$ *Escura2*, $\{$*Escura1, Escura2, Lilás*$\}$ $\preceq$ *Escura*.

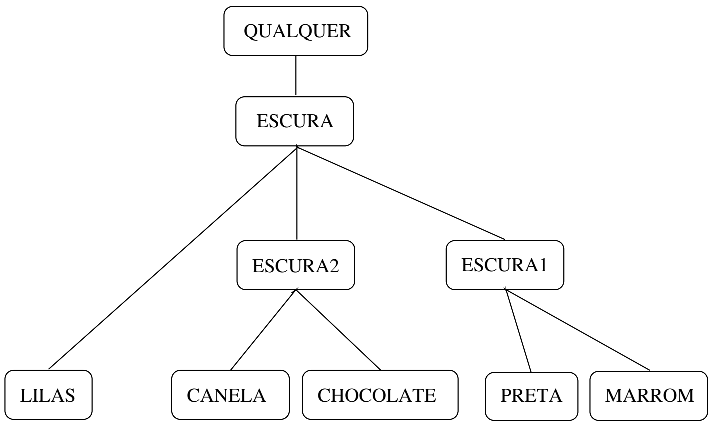

Figura 2: Hierarquia de conceitos para os atributos cujo domínio seja *cor*

Hieraquia sobre valores de atributos Obtida através de operações sobre os valores de atributos como codificação ou extração de informação, como no caso de atributos

numéricos, em que faixas de valores são estabelecidas agrupando as tuplas e discretizando os valores encontrados. Neste caso, as notas atribuídas aos alunos de pós-graduação numa universidade poderiam ser agrupadas nas seguintes faixas:
$\{0.0 \ldots 4.9\} \preceq$ *Insuficiente,* $\{5.0 \ldots 7.0\} \preceq$ *Bom,* $\{7.1 \ldots 8.5\} \preceq$ *Muito Bom,* $\{8.6 \ldots 10.0\} \preceq$ *Excelente.*

Hierarquia baseada em regras Definida por relações mais complexas. Em alguns casos, pode ser necessário representar uma generalização de conceitos que dependa não apenas do valor, mas também de alguma condição imposta sobre valores de outros atributos. Por exemplo, no ítem anterior, a nota 6.5 poderia ser *Muito Bom*, caso o aluno fosse da graduação, e então a hierarquia seria:
$\{6.5 \ldots 8.5\} \wedge \{graduação\} \preceq$ *Muito Bom.*

O armazenamento de hierarquias de conceitos pode ser feito através de arquivos ou de tabelas relacionais. A vantagem do uso de tabelas relacionais resulta das facilidades fornecidas pelo Sistema Gerenciador de Banco de Dados na geração de índices, no uso concorrente de uma mesma tabela e na segurança dos dados [Lu, 1997].

2.3. Medidas de Relevância

Existem diversas medidas de relevância de padrões descobertos. Dentre as mais usadas estão as medidas de suporte e confiança [Agrawal et al., 1993] [Agrawal and Srikant, 1995], que servem como exemplo para mostrar as vantagens da descoberta em múltiplos níveis. O *suporte* de uma regra $X \rightarrow Y$ é a razão entre o número de tuplas que contém o antecedente e o conseqüente da regra e o número de tuplas que pertence àquela classe, i.e., a probabilidade que a regra ocorra dentro da classe. A *confiança* é a razão entre o número de tuplas que contém o antecedente e o conseqüente da regra e o número de tuplas em que ocorre o antecedente da regra, i.e., a probabilidade condicional do conseqüente de uma regra dado que ocorre o antecedente: $P(Y|X)$. Regras com altos valores de suporte e confiança são denominadas regras fortes.

2.4. Representação e armazenamento de Hierarquias do Conceitos

Uma hierarquia de conceitos pode ser representada como uma árvore onde cada vértice representa um conceito e cada aresta uma relação de especialização-generalização entre os conceitos dos vértices adjacentes. Neste árvore, o vértice que possui o mais alto grau de abstração denomina-se vértice raiz, que denominamos de *QUALQUER*. Os vértices cujos conceitos são valores de atributos nas bases de dados são denominados vértices folha, não havendo conceitos mais específicos que estes.

0	*1*	*2*	*3*
QUALQUER	Escura	Escura1	Marrom
QUALQUER	Escura	Escura1	Preta
QUALQUER	Escura	Escura2	Canela
QUALQUER	Escura	Escura2	Chocolate
QUALQUER	Escura	Lilás	
QUALQUER	Clara	Branca	
QUALQUER	Clara	Rosa	
QUALQUER	Clara	Amarela	
QUALQUER	Clara	Laranja	

Tabela 1: Hierarquia de conceitos sobre *cor* codificada numa tabela relacional

Podemos armazenar uma hierarquia de conceitos através de uma tabela relacional, onde todos os caminhos entre o vértice raiz e cada vértice folha são armazenados como tuplas e cada atributo desta tabela contém os conceitos de um mesmo nível. Através da tabela 1, observa-se como a hierarquia da figura 2 é representada numa tabela relacional. Na figura 2 os conceitos Marrom, Preta, Canela, Chocolate, Lilás, Branca, Amarela, Laranja e Rosa são valores do atributo na base de dados, e na árvore, serão vértices folha.

Esta hierarquia poderá ser utilizada para todos os atributos que possuam seus valores neste domínio. Na tabela 2, mostramos algumas tuplas da tabela que contém dados sobre cogumelos. Os atributos *cor esporo* e *cor do topo* possuem valores dentro do domínio da hierarquia da figura 2 e esta hierarquia poderá ser utilizada para generalizar os conceitos contidos nos vértices folha.

cor_esporo	*forma do topo cogumelo*	*cor_topo*	*classe*
lilás	cônica	rosa	comestível
preta	achatada	canela	venenoso
amarela	convexa	branca	comestível
amarela	sino	rosa	venenoso
marrom	achatada	canela	venenoso
lilás	convexa	branca	comestível

Tabela 2: Extrato da tabela relacional com atributos sobre cogumelos

3. Descrição do Algoritmo de Classificação

O algoritmo de regras de classificação em múltiplos níveis proposto neste trabalho foi baseado em outros dois algoritmos [Taylor, 1999, Fu, 1996], combinando suas características. A seguir especificamos e descrevemos o algoritmo passo a passo.

Entrada: Um banco de dados e hierarquias de conceitos para cada atributo.
Saída: Conjunto das regras obtidas.
Método:

1. seleção dos atributos, das hierarquias de conceitos de cada atributo, do atributo classificador e dos valores mínimos de suporte e confiança;
2. gerar os conjuntos de co-ocorrências para cada atributo;
3. gerar as hipóteses de regras que contêm apenas um par atributo-valor e armazená-las em H;
4. para cada $h \in H$
    se valores de suporte e confiança de h são maiores que valores mínimos **então**:
        armazena h como regra descoberta removendo-a de H;
    senão
        se suporte maior que valor mínimo **então**
            refinar h e armazenar em H;

3.1. Descrição

O conjunto de dados onde será realizada a mineração deve estar representado como uma única tabela, i.e., caso os dados estejam representados em diversas tabelas, eles devem ser unidos numa única tabela, diferentemente do algoritmo proposto em [Fu, 1996].

Para a execução da mineração dos dados, as tabelas com as hierarquias de conceitos são lidas e armazenadas numa estrutura em árvore, na memória principal, através de listas de adjacência. Esta representação em memória visa reduzir o tempo para percorrer os conceitos da hierarquia, que é executado diversas vezes para obtenção dos vértices folha. Mais de uma hierarquia pode ser utilizada para um mesmo atributo de uma mesma tabela, o que não acontece nos algoritmos citados anteriormente.

A determinação das regras de classificação pode ser vista como uma busca através de um espaço de regras candidatas. Esta busca pode ser iniciada com uma regra vazia e, sucessivamente, procurar por regras com um atributo, dois atributos, três atributos até, possivelmente, uma regra que contenha todos os atributos. O tamanho deste espaço de busca pode ser determinado da seguinte forma: se cada tupla possuir i atributos com k possíveis valores para cada um (incluindo os valores das hierarquias de conceitos de cada atributo), há k^i possíveis combinações de tuplas. Qualquer subconjunto dos atributos pode corresponder a uma regra o que significa 2^{k^i} regras possíveis. Como podemos ver, o tamanho deste espaço de busca é suficientemente grande para tornar o problema intratável. Logo, faz-se necessário diminuí-lo através de funções heurísticas. Uma delas é o uso das co-ocorrências entre atributos, proposto em [Taylor, 1999] fazendo com que apenas os valores de atributos que ocorrem nas tuplas da base de dados possam ser utilizados como regras candidatas, ao invés de tentar combinações inexistentes na base de dados. Outra forma é não avaliar regras que possuam pares atributo-valor que, por si só, já sejam regras com valores mínimos de suporte e confiança. Aqui, o acréscimo de pares atributo-valor à uma regra já descoberta, apenas envolve um subconjunto das tuplas desta regra, não trazendo qualquer novidade.

Seja H o conjunto de regras candidatas, ou hipóteses, que serão verificadas pelo algoritmo e, h o elemento deste conjunto que está sendo verificado correntemente pelo algoritmo. Cada elemento h de H, inicialmente, é formado por apenas um par atributo-valor, cujo valor será o do nível conceitual imediatamente abaixo de *QUALQUER*, ou seja, o segundo nível conceitual, já que *QUALQUER* reúne todos os valores possíveis do atributo na base de dados de uma determinada hierarquia. Um atributo que possua mais de uma hierarquia, irá gerar tantas regras quanto o número de hierarquias que possui. Caso seja realizada alguma substituição do valor de um par atributo-valor, esta será feita por um conceito descendente na respectiva hierarquia, utilizando uma abordagem de cima para baixo, i.e., *top-down*. A cada iteração do algoritmo, as hipóteses examinadas conterão 1, 2, 3, $\ldots$, n pares atributo valor. Em geral, na iteração i, as hipóteses de regras terão i pares atributo-valor. As operações de verificação das hipóteses de regras ocorrem enquanto houver elemento em H e uma das duas condições for satisfeita:

Condição (a) o valor de suporte e confiança de h é maior que os valores mínimos determinados;

Condição (b) o valor de suporte de h é maior que o valor mínimo determinado e o valor de confiança é menor que o valor mínimo.

Se a condição (a) for satisfeita, então a hipótese é transformada em regra e h é retirado de H. Se a condição (b) for satisfeita, a hipótese será refinada na tentativa de aumentar a confiança. Se ambas condições forem falsas, esta hipótese é descartada e retirada do conjunto H.

3.2. Refinando hipóteses de regras

O processo de busca ocorre como uma *expansão de vértices sucessores*, ou seja, o refinamento da hipótese pode ser feito de duas maneiras:

(i) especializando conceitualmente a regra do vértice a ser refinado (expandido) ou

(ii) ou adicionando-lhe mais um par atributo-valor pertencente ao conjunto das co-ocorrências do último par da regra.

O refinamento de uma regra h se dá, inicialmente, através da especialização conceitual. Esta especialização é realizada no último par atributo-valor da hipótese, gerando um conjunto de hipóteses *filhas* denominado H_f. O conjunto H_f contém todas as possíveis regras filhas, indo do conceito atual até os valores mais específicos, i.e., os vértices folha da hierarquia de conceitos. Como numa busca em largura, para cada classe, cada $h_f \in H_f$

tem seus valores de suporte e confiança testados. Se os valores são maiores ou iguais aos mínimos, então esta hipótese é transformada em regra e conceitos mais específicos não são verificados. Conceitos mais específicos só serão verificados enquanto: o valor de suporte for maior ou igual que o mínimo e o valor de confiança for menor que o mínimo. Se em algum momento o valor de suporte for menor que o mínimo, esta hipótese é descartada e valores conceitualmente mais específicos não serão verificados. Ao verificar regras cujos conceitos são valores encontrados no banco de dados, i.e., vértices folha na hierarquia, que não podem ser mais especializados, o algoritmo irá refinar a regra adicionando-lhe mais um par atributo-valor.

O refinamento de uma regra sempre reduz seu valor de suporte no banco de dados. Um conceito mais específico contém menos exemplos num banco de dados do que um conceito que seja seu ascendente pois, o ascendente pode englobar outros exemplos da base de dados que pertençam ao conjunto de um outro descendente seu diferente do primeiro. Do mesmo modo, a adição de um outro par atributo-valor terá como resultado a intersecção dos exemplos contidos em cada par atributo-valor individualmente, pois se trata de uma conjunção de pares atributo-valor, a menos que os dois conjuntos tenham os mesmos elementos.

Já o valor de confiança poderá ser maior. Um conceito descendente numa hierarquia poderá conter, no banco de dados, elementos que pertençam mais a uma classe do que outra, ao contrário de um outro conceito irmão. Ocorre o mesmo no acréscimo de um outro par atributo valor onde, os exemplos resultantes da intersecção, poderão estar contidos mais numa mesma classe do que noutra.

Adicionando-se um novo par atributo-valor gera-se uma nova hipótese h' que será inserida em H e, o exame desta nova hipótese será realizado após todas as anteriores, como numa fila, garantindo que a cada iteração i apenas hipóteses de regras contendo i pares atributo-valor são examinadas.

3.3. Corte no espaço de busca

Para fazer cortes no espaço de busca, é feita ainda uma análise adicional das hipóteses (não descrita no algoritmo acima por motivo de clareza) que discutimos a seguir.

Como visto no item anterior, no processo de refinamento uma hipótese h pode ser (i) especializada conceitualmente, segundo a hierarquia de conceitos, ou (ii) estendida através da adição de mais um par atributo-valor. O algoritmo adota a hipótese da *navalha de Occam*: uma regra mais simples é melhor. Uma regra é mais simples que outra se possuir menor número de pares atributo-valor. Assim, o processo de especialização conceitual é realizado antes da adição de um novo par atributo-valor, e regras descobertas não serão combinadas com outros atributos para gerar novas regras que sejam maiores, i.e., regras *menos simples*.

Há três condições relacionadas a uma hipótese h e suas hipóteses *filhas* h_f que são avaliadas pelo algoritmo:

Condição 1 um conjunto de regras filhas onde existe uma que cobre, pelo menos, 90% das tuplas cobertas pela regra pai e possui o valor de suporte maior que o mínimo.

Condição 2 um conjunto de regras filhas, onde todas possuem os valores de suporte inferiores ao valor mínimo determinado.

Condição 3 um conjunto de regras filhas onde existe pelo menos uma que possui suporte maior que o valor mínimo e cobre menos que 90% das tuplas cobertas pela regra pai.

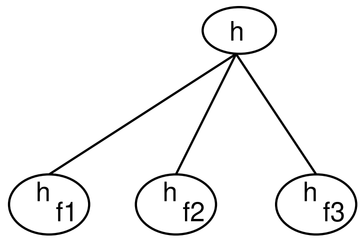

Figura 3: Refinamento de uma hipótese de regra através da especialização conceitual do valor do atributo

Seja h_{f1}, como mostrado na figura 3, uma hipótese filha de h com a condição **1**. Neste caso, a hipótese h é muito geral e a hipótese h_{f1}, que é uma especialização conceitual, é menos geral. Conseqüentemente, h_{f1} será usada para gerar regras que cobrem tuplas também cobertas por h. Se a hipótese h_{f1} possuir os valores mínimos de suporte e confiança, ela será armazenada como regra descoberta e não a hipótese h. Caso nenhuma das hipóteses *filhas* tenha o valor de confiança maior ou igual ao mínimo, o algoritmo irá refinar esta hipótese adicionando-lhe mais um par atributo-valor. Como h_{f1} cobre 90% das tuplas cobertas por h, a adição de um novo par atributo-valor será feita à hipótese de regra h_{f1} e não a h. Ao contrário, se a condição **1** for falsa, a adição de um novo par atributo-valor será feita em h.

Se a condição **2** for verdadeira, todas as hipóteses filhas são descartadas e o refinamento se dará através da adição de um novo par atributo-valor à hipótese h.

Se a condição **3** for verdadeira todas as hipóteses filhas são descartadas e o refinamento se dará através da adição de um novo par atributo-valor à hipótese h. Esta condição conduz a seguinte questão. Não é garantido que as regras filhas poderão gerar regras com medidas acima dos valores mínimos. Decidindo por manter h e não utilizar uma hipótese *filhas* para a adição de um par atributo-valor, mantêm-se as tuplas que são cobertas por h e diminui-se a redução do suporte sofrida pelo acréscimo de mais um par atributo-valor. O valor de 90% foi obtido empiricamente após alguns testes no banco de dados utilizado. Com valores inferiores a 90% das tuplas cobertas pela hipótese *pai*, as hipóteses *filhas* não resultaram em regras válidas pois, o valor de suporte da hipótese estendida, i.e., com mais um par atributo-valor, foi inferior ao valor mínimo determinado pelo usuário. Isto é devido à redução no número de tuplas cobertas por uma hipótese refinada, como comentado anteriormente.

As medidas de relevância podem ser ajustadas pelo usuário para limitar as regras descobertas àquelas que possuem apenas valores superiores aos mínimos desejados o que, apesar de não buscar por todas as combinações possíveis, reduz o espaço de busca a regras que possuam maior relevância.

3.4. Otimizando as consultas no Banco de Dados

Ao efetuar a busca pelas tuplas que são cobertas pelo antecedente de uma hipótese de regra que está descrita através de um conceito de alto nível, os valores procurados são os valores folha das hierarquias de conceitos pois são estes que estão registrados no BD. Desta forma, uma pesquisa no BD de uma regra de alto nível também lê os conceitos nos níveis inferiores. O refinamento de uma hipótese de regra se dá inicialmente através da especialização conceitual do valor de atributo. Como todos os valores conceitualmente inferiores também são lidos ao ser buscado um valor de alto nível, com apenas uma busca ao BD o algoritmo já possui os dados necessários para calcular as medidas de relevância de valores de mais baixo nível. Desta forma, apenas uma consulta ao BD por regra é suficiente para o algoritmo calcular as medidas de relevância para cada conceito da hierarquia, e reduz-se o acesso ao disco onde está armazenado o BD.

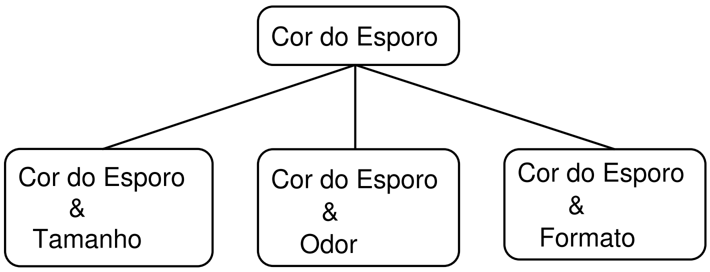

Figura 4: Refinamento de uma hipótese de regra através da adição de um novo par atributo-valor

Quando uma hipótese de regra é refinada adicionando-se um novo par atributo-valor, como mostrado na figura 4, as tuplas que o antecedente da regra *cor do esporo & odor* cobre pertencem ao conjunto das tuplas cobertas pelo antecedente da regra não estendida, *cor do esporo*, ou seja, a hipótese de regra antes de receber o novo par atributo-valor, pois agora o conjunto de tuplas será formado pela intersecção das tuplas que são cobertas por cada par atributo-valor que integra a nova regra. Ao invés de buscar as tuplas de uma nova regra estendida diretamente na base de dados, será melhor buscá-las no conjunto das tuplas formado pela regra antes desta ser estendida, diminuindo o número de tuplas que serão lidas pelo algoritmo para verificar as medidas de relevância. Isto conduz a idéia de armazenar o conjunto de tuplas que é coberto pelo antecedente de cada regra antes de adicionar-lhe um novo par.

O valor do par atributo-valor adicionado à uma hipótese de regra será o valor de mais alto nível conceitual imediatamente abaixo de *QUALQUER* na hierarquia de conceitos deste atributo. Em vez de efetuar um *JOIN* entre as tuplas cobertas pelo antecedente da regra antes de ser estendida e a tabela de dados, as tuplas deste novo par atributo-valor podem estar armazenadas numa outra tabela, e agora o *JOIN* é efetuado entre duas tabelas que contêm menos tuplas do que toda a tabela de dados. Ao ser iniciado o algoritmo, todos os atributos que foram selecionados são utilizados para gerar hipóteses de regras. Deste modo, antes de começar a verificação das medidas de relevância das hipóteses de regra, para cada atributo-valor de mais alto nível é criada uma tabela contendo as tuplas que possuem os valores do atributo em questão. Como o algoritmo pode utilizar mais de uma hierarquia para um mesmo atributo, será criada uma tabela para cada hierarquia de conceito relacionada ao atributo.

As tabelas que armazenam as tuplas do antecedente de uma regra possuem apenas uma coluna contendo o atributo identificador da tupla, que é único na tabela de dados. Já as tabelas dos atributos contêm, além do identificador da tupla, que será utilizado no *JOIN*, o valor do atributo no banco de dados e a classe a que pertence.

3.5. Diagrama funcional do Algoritmo

Na figura 5 é apresentado o diagrama funcional do algoritmo de descoberta.

A seleção de atributos é feita pelo usuário que escolhe quais atributos o algoritmo irá gerar as hipótese de regras. A partir dos atributos são geradas as listas de co-ocorrência de cada atributo selecionado. O gerador de hipóteses recebe como entrada os atributos selecionados, as hierarquias de conceitos de cada um dos atributos selecionados e o critério de poda, como descrito no item 3.3. As regras descobertas irão evitar que uma regra já descoberta venha a ser testada novamente. Isso pode ocorrer quando são buscadas regras com mais de um par atributo-valor. Como o antecedente é uma conjunção de pares atributo-valor, regras que possuem o mesmo conjunto de pares como antecedente são regras iguais, independentemente

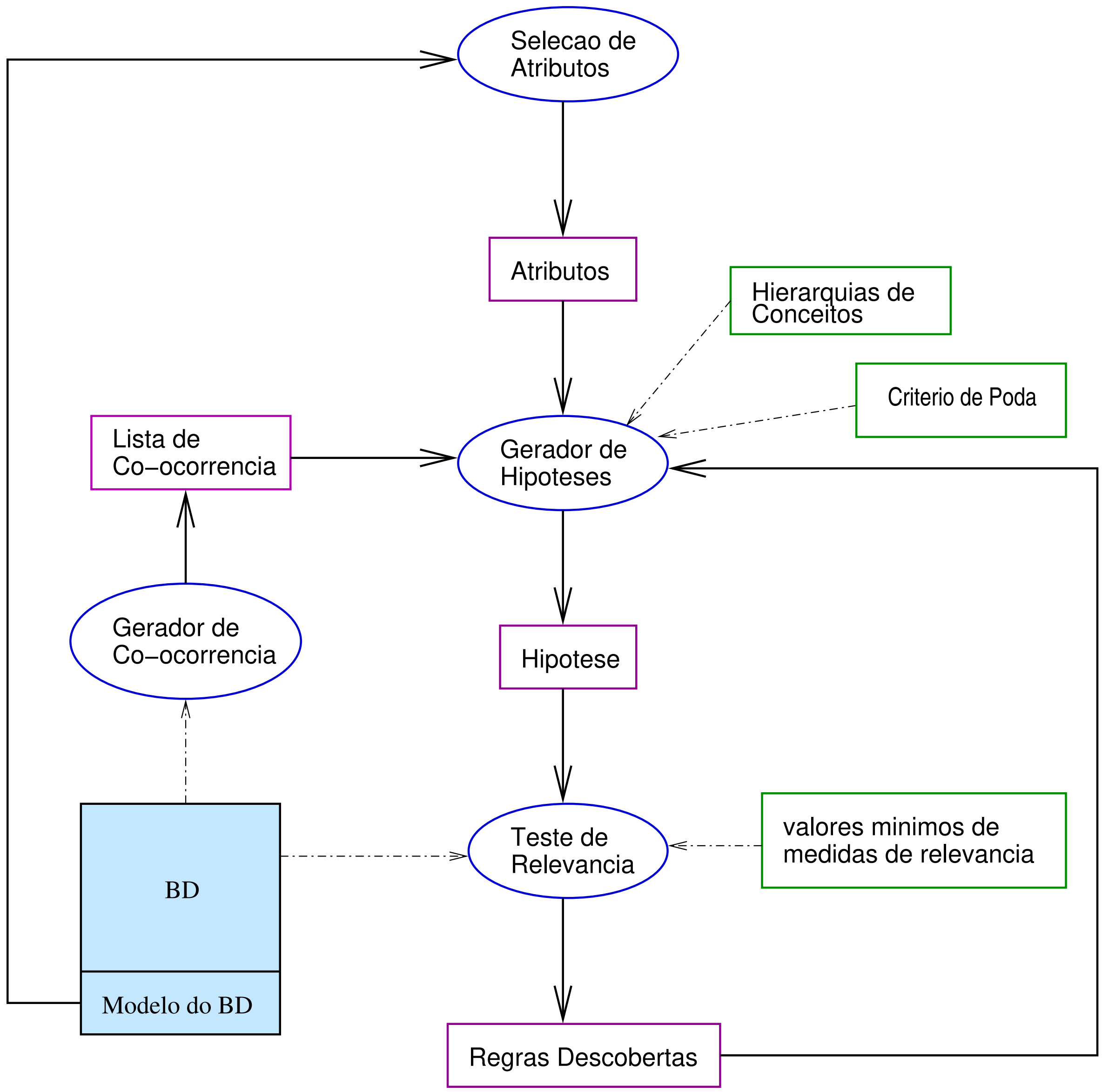

Figura 5: Diagrama funcional do Algoritmo

da ordem dos pares. Logo, o gerador não deverá buscar por regras já descobertas. Após ser gerada uma hipótese de regra, ela é testada a fim de verificar-se as medidas de relevância. Para calcular as medidas de relevância de determinada hipótese de regra, o algoritmo se comunica com o banco de dados através de expressões SQL. Se a hipótese de regra possuir os valores mínimos de relevância, ela é armazenada como regra descoberta. Caso contrário, esta hipótese deverá ser refinada, se for possível, pelo gerador de hipóteses para uma nova verificação.

4. Descoberta de conhecimento no banco de dados sobre cogulemos

Utilizando a tabela 2, sobre cogumelos, queremos classificar as tuplas em **comestível** ou **venenoso**. Durante o processo de busca, são feitas consultas ao BD para obter as medidas de suporte e confiança. Por exemplo, a expressão em SQL para obter o número de tuplas em cada classe da regra **se** cor topo = clara **então** classe é comestível ou venenoso será:

```
SELECT CLASSE, COUNT(*)
FROM TABELA
WHERE COR_TOPO=´ROSA´ OR COR_TOPO=´AMARELA´ OR COR_TOPO=´BRANCA´
        OR COR_TOPO=´LARANJA’
GROUP BY CLASSE;
```

Ainda na mesma tabela, regrashipóteses de regras que possuam mais de um par atributo-valor como **se** cor topo = clara e cor esporo = escura1 **então** classe é comestível ou venenoso terá a seguinte expressão em SQL:

```
SELECT CLASSE, COUNT(*)
FROM TABELA
WHERE (COR_TOPO=´ROSA´ OR COR_TOPO=´AMARELA´ OR COR_TOPO=´BRANCA´
        OR COR_TOPO=´LARANJA’) AND (COR_ESPORO=´PRETA’
        OR COR_ESPORO=´MARROM´)
GROUP BY CLASSE;
```

Note que a construção da expressão é feita automaticamente pelo algoritmo proposto, consultando as hierarquias conceituais correspondentes aos atributos do BD e montando a expressão disjuntiva do campo *WHERE*.

Para cada base de dados, há uma tabela que armazena os dados das hierarquias desta base, contendo, em cada tupla, o nome de uma tabela da base de dados, um atributo desta tabela, o nome da tabela que contém uma hierarquia para este atributo e o número de vértices existentes nesta hierarquia. Podemos ter mais de uma hierarquia para um mesmo atributo de uma tabela qualquer bem como uma mesma hierarquia poderá ser utilizada para vários atributos, desde que seu domínio esteja contido nos conceitos da hierarquia.

4.1. Regras descobertas

A tabela 3 apresenta as regras descobertas utilizando o banco de dados sobre cogumelos. O símbolo $\uparrow$ significa que o conceito do atributo não está presente no banco de dados e é um conceito de alto nível, obtido através da hierarquia de conceitos do atributo. O símbolo $\rightarrow$ indica o conseqüente da regra, ou seja, a classe cujos valores podem ser **e**, que significa venenoso ou *edible* em inglês e **p**, que significa comestível ou *poisonous*.

5. Sistemas Utilizados

O Sistema Gerenciador de Banco de Dados utilizado é o PostgreSQL 7.3.2 e a linguagem de programação empregada é Java. A interação com o sistema é feita através de uma interface visual que permite selecionar a base de dados desejada dentre as existentes no SGBD. Após isso, pode-se escolher a tabela e os atributos que irão compor as regras, bem como o atributo classe. Há uma opção de seleção das hierarquias de conceitos que serão utilizadas para cada atributo da tabela selecionada e pode-se estabelecer os valores de suporte e confiança para extração das regras de classificação.

A base de dados utilizada foi obtida do repositório de bases de dados de aprendizado de máquina da Universidade da Califórnia, Irvine, EUA. Foi escolhida uma base de dados que contém 8416 tuplas descrevendo cogumelos com 23 atributos utilizados para descrever duas classes de cogumelos, comestíveis ou venenosos.

6. Conclusões

Foi apresentado um algoritmo para descoberta de regras de classificação utilizando conhecimento prévio representado através de hierarquias de conceitos. Diferentemente de [Fu, 1996], a abordagem de cima para baixo *(top-down)* permite descobrir regras com maior relevância e possivelmente mais interessantes. Também não utilizamos tabelas generalizadas para reduzir o espaço de busca como em [Han et al., 1992]. Ao invés disso, após serem escolhidos os atributos, são gerados os conjuntos de co-ocorrências para reduzir o espaço de busca. O uso de um SGBD relacional como meio de armazenamento da base de dados e das hierarquias de conceitos, flexibiliza o emprego do algoritmo em qualquer banco de dados deste tipo, ao contrário de [Taylor, 1999], e a possibilidade de utilizar mais de uma hierarquia por atributo, permite ao usuário avaliar regras obtidas a partir de diferentes especificações de conceitos sem precisar executar um mesmo algoritmo várias vezes para cada hierarquia.

NR	*Descrição da regra*	*suporte*	*confiança*
1	odor = n $\rightarrow$ e	0.8098859	0.9659864
2	odor $\uparrow$ bad $\rightarrow$ p	0.8222676	1.0
3	gill color = b $\rightarrow$ p	0.4412666	1.0
4	ring type = l $\rightarrow$ p	0.33094996	1.0
5	spore print color = h $\rightarrow$ p	0.40449437	0.9705882
6	gill size = n & gill color $\uparrow$ light $\rightarrow$ p	0.49846783	0.92424244
7	gill size = n & stalk color below ring $\uparrow$ light $\rightarrow$ p	0.5638406	0.9019608
8	gill size = n & ring type = e $\rightarrow$ p	0.4514811	0.97356826
9	gill size = n & spore print color = w $\rightarrow$ p	0.453524	0.9736842
10	gill size = n & population = v $\rightarrow$ p	0.5066394	0.90181816
11	gill size = b & gill color $\uparrow$ dark1 $\rightarrow$ e	0.2851711	1.0
12	gill size = b & ring type = e $\rightarrow$ e	0.22813688	1.0
13	gill size = b & ring type = l $\rightarrow$ p	0.33094996	1.0
14	gill size = b & spore print color = h $\rightarrow$ p	0.40449437	1.0
15	gill size = b & spore print color $\uparrow$ dark1 $\rightarrow$ e	0.7718631	1.0
16	stalk color below ring $\uparrow$ light & ring type = l $\rightarrow$ p	0.2206333	1.0
17	stalk color below ring $\uparrow$ light & spore print color = h $\rightarrow$ p	0.29417774	0.96
18	ring type $\uparrow$ type3 & spore print color = h $\rightarrow$ p	0.40449437	1.0
19	ring type = e & population = v $\rightarrow$ p	0.4494382	0.9734513
20	spore print color = w & population = v $\rightarrow$ p	0.4494382	0.9649123
21	spore print color $\uparrow$ dark1 & population = y $\rightarrow$ e	0.23954372	1.0
22	gill size = b & stalk color below ring $\uparrow$ light & gill color $\uparrow$ dark1 $\rightarrow$ e	0.25095057	1.0
23	gill size = b & stalk color below ring $\uparrow$ light & ring type = e $\rightarrow$ e	0.20532319	1.0
24	gill size = b & ring type = p & gill color $\uparrow$ dark $\rightarrow$ e	0.29657796	0.9285714
25	gill size = b & ring type = p & population $\uparrow$ non centralized $\rightarrow$ e	0.35361215	0.9117647
26	gill color $\uparrow$ light & stalk color below ring $\uparrow$ light & gill size = n $\rightarrow$ p	0.49438202	0.9453125
27	gill color $\uparrow$ light & ring type = e & gill size = n $\rightarrow$ p	0.4514811	0.97356826
28	gill color $\uparrow$ light & ring type = e & spore print color = w $\rightarrow$ p	0.4514811	0.9246862
29	gill color $\uparrow$ light & ring type = e & population = v $\rightarrow$ p	0.4494382	0.9734513
30	gill color $\uparrow$ light & spore print color = w & gill size = n $\rightarrow$ p	0.453524	0.9736842
31	gill color $\uparrow$ light & spore print color = w & population = v $\rightarrow$ p	0.4494382	0.9649123
32	gill color $\uparrow$ light & population = v & gill size = n $\rightarrow$ p	0.47191012	0.9314516
33	gill color $\uparrow$ dark & spore print color $\uparrow$ dark1 & gill size = b $\rightarrow$ e	0.42205322	1.0
34	stalk color below ring $\uparrow$ light & ring type = e & gill size = n $\rightarrow$ p	0.4473953	1.0
35	stalk color below ring $\uparrow$ light & ring type = e & spore print color = w $\rightarrow$ p	0.4473953	0.9480519
36	stalk color below ring $\uparrow$ light & ring type = e & population = v $\rightarrow$ p	0.4453524	1.0
37	stalk color below ring $\uparrow$ light & spore print color $\uparrow$ dark1 & gill size = b $\rightarrow$ e	0.634981	1.0
38	stalk color below ring $\uparrow$ light & spore print color = w & gill size = n $\rightarrow$ p	0.4494382	1.0
39	stalk color below ring $\uparrow$ light & spore print color = w & population = v $\rightarrow$ p	0.4453524	0.9909091
40	stalk color below ring $\uparrow$ light & population = v & gill size = n $\rightarrow$ p	0.50255364	0.92134833
41	ring type $\uparrow$ type3 & spore print color $\uparrow$ dark1 & gill size = b $\rightarrow$ e	0.5893536	1.0
42	ring type $\uparrow$ type3 & spore print color $\uparrow$ dark1 & population = y $\rightarrow$ e	0.23954372	1.0
43	ring type = e & spore print color = w & gill size = n $\rightarrow$ p	0.4514811	0.97356826
44	ring type = e & spore print color = w & population = v $\rightarrow$ p	0.4494382	0.9734513
45	spore print color $\uparrow$ dark1 & population $\uparrow$ centralized & gill size = b $\rightarrow$ e	0.3688213	1.0
46	gill size = b & ring type = p & stalk color below ring $\uparrow$ light & gill color $\uparrow$ dark $\rightarrow$ e	0.22813688	0.90909094
47	gill color $\uparrow$ light & stalk color below ring $\uparrow$ light & ring type = e & gill size = n $\rightarrow$ p	0.4473953	1.0
48	gill color $\uparrow$ light & stalk color below ring $\uparrow$ light & ring type = e & spore print color = w $\rightarrow$ p	0.4473953	0.97333336
49	gill color $\uparrow$ light & stalk color below ring $\uparrow$ light & ring type = e & population = v $\rightarrow$ p	0.4453524	1.0
50	gill color $\uparrow$ light & stalk color below ring $\uparrow$ light & spore print color = w & gill size = n $\rightarrow$ p	0.4494382	1.0
51	gill color $\uparrow$ light & stalk color below ring $\uparrow$ light & spore print color = w & population = v $\rightarrow$ p	0.4453524	0.9909091
52	gill color $\uparrow$ light & stalk color below ring $\uparrow$ light & population = v & gill size = n $\rightarrow$ p	0.4678243	0.95416665
53	gill color $\uparrow$ dark & spore print color $\uparrow$ dark1 & stalk color below ring $\uparrow$ light & gill size = b $\rightarrow$ e	0.35361215	1.0
54	ring type $\uparrow$ type3 & spore print color $\uparrow$ dark1 & stalk color below ring $\uparrow$ light & gill size = b $\rightarrow$ e	0.4524715	1.0
55	ring type $\uparrow$ type3 & spore print color $\uparrow$ dark1 & population $\uparrow$ centralized & gill size = b $\rightarrow$ e	0.27756655	1.0

Tabela 3: Regras descobertas do banco de dados sobre cogumelos com suporte na classe $\geq$ 10% e confiança $\geq$ 90%

6.1. Tópicos para Pesquisa

Após o desenvolvimento deste sistema de descoberta, alguns tópicos de pesquisa se tornam relevantes para melhorar, não só o algoritmo em si, mas também o seu resultado, isto é, os padrões descobertos.

Uma tupla no banco de dados pode ser representada através de uma grande quantidade de atributos, por exemplo, uma cadeida genética poderá conter 4000 atributos que serão utilizados na tarefa de classificação. Isso aumenta consideravelmente o número de regras a serem testadas. Entretanto, nem todos os atributos de uma tupla tem relevância na descoberta de um padrão. Alguns atributos não possuem qualquer correlação com a classe da tupla e podem ser desconsiderados por ocasião da geração de hipóteses de regras. Logo, faz-se necessário selecionar as características que melhor atendam a classificação de determinada base de dados. Algumas abordagens como por exemplo, o cálculo da entropia de cada atributo da base de dados, podem ajudar a reduzir o número de atributos que serão selecionados para a execução do algoritmo.

As hierarquias utilizadas representam apenas relações de generalização e especialização simples, i.e., com apenas um caminho entre dois vértices quaisquer. Entretanto, há relações de generalização-especialização que não obedecem esta característica, e isso conduz ao estudo de como representar e utilizar hierarquias deste tipo, bem como ferramentas que possam responder a requisições sobre estas relações para serem utilizadas num banco de dados.

As medidas de relevância utilizadas, suporte e confiança, são medidas simples mas que, em algumas bases de dados, não são adequadas. Como apresentado por [Kodratoff, 2001], há outras medidas que fornecem resultados relevantes e que também devem ser testadas na base de dados.

Referências

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Mining Association Rules between Sets of Items in Large Databases*, Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, EUA.

Agrawal, R. and Srikant, R. (1995). Mining generalized association rules. In *Mining Generalized Association Rules*, Proceedings of 21st VLDB Conference, Zurique, Suíça.

Fayyad, U. M., Piatesky-Shapiro, G., and Smyth, P. (1996). *Advances in Knowledge Discovery and Data Minig*. MIT Press.

Fu, Y. (1996). *Discovery of Multiple-Level Rules from Large Databases*. Phd thesis, School of Computer Science of Simon Fraser University, Burnaby, B.C., Canadá.

Han, J., Cai, Y., and Cercone, N. (1992). Knowledge discovery in databases: An attribute-oriented approach. In *Proceedings of the 18th VLDB Conference*, Vancouver, British Columbia, Canadá.

Kodratoff, Y. (2001). Comparing machine learning and knowledge discovery in databases: An application to knowledge discovery in texts. In *Artificial Intelligence*, pages 1–21. Springer Verlag.

Lu, Y. (1997). Concept hierarchy in data mining: Specification, generation and implementation. Masters thesis, School of Computer Science of Simon Fraser University, Burnaby, B.C., Canadá.

Taylor, M. G. (1999). *Finding High Level Discriminant Rules in Parallel*. Phd thesis, Faculty of the Graduate School of the University of Maryland, College Park, EUA.