UNIVERSIDADE DO EXTREMO SUL CATARINENSE
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JHONAS BONFANTE GUINZANI

MINERAÇÃO DE DADOS EM REDES BAYESIANAS UTILIZANDO A API DA *SHELL BELIEF NETWORK POWER CONSTRUCTOR* (BNPC)

CRICIÚMA, JULHO DE 2006.

JHONAS BONFANTE GUINZANI

MINERAÇÃO DE DADOS EM REDES BAYESIANAS UTILIZANDO A API DA *SHELL BELIEF NETWORK POWER CONSTRUCTOR* (BNPC)

Trabalho de Conclusão de Curso para a Obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientadora: Prof. M.Sc. Priscyla Waleska T. A. Simões

CRICIÚMA, JULHO DE 2006.

JHONAS BONFANTE GUINZANI

MINERAÇÃO DE DADOS EM REDES BAYESIANAS UTILIZANDO A API DA *SHELL BELIEF NETWORK POWER CONSTRUCTOR* (BNPC)

Submetido ao corpo docente do Departamento de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

_____

Profª. M.Sc. Ana Cláudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

_____

Profª. M.Sc. Priscyla Waleska T. A. Simões (UNESC)
Orientadora

_____

Profª. M.Sc. Merisandra Côrtes de Mattos (UNESC)

_____

Prof. M.Sc. Maurício de Paula (UNISUL)

Aos meus pais, Dilcionir e Marli, minha irmã Maria Letícia e meus amigos pelo incentivo, apoio e compreensão.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me guiar e iluminar nos momentos difíceis, me impulsionando a enfrentar os desafios e a seguir em frente na realização dos meus objetivos.

À minha família por ser a minha base e por estar sempre ao meu lado, ajudando, aconselhando e principalmente me dançó força nos momentos que eu preciso.

Aos meus amigos, em especial ao Adilson, Eduardo, Elaine e Lilia, pela amizade, compreensão, palavras e incentivos durante essa caminhada.

Aos meus colegas de graduação pelos momentos bons vividos durante esses anos.

Aos professores do Curso de Ciência da Computação por terem nos transmitido seus conhecimentos.

E finalmente, a minha orientadora Priscyla, que esteve ao meu lado, contribuindo com seu conhecimento, orientação e incentivo.

RESUMO

O grande volume de informações nos bancos de dados atuais torna difícil a análise dos dados. Essa quantidade de informação pode esconder relações significativas somente sendo encontradas por técnicas inteligentes apropriadas. Dessa forma, o processo de Descoberta de Conhecimento em Bases de Dados (DCBD) reúne tarefas e métodos para a extração de conhecimento relevante dessas bases, por meio de sua etapa principal, a de mineração de dados, que pode ser realizada de diversas maneiras, destacando-se a fundamentada em redes bayesianas. Na realização desta pesquisa foram analisadas as *Application Programming Interfaces (*API) das *shells* de mineração de dados em redes bayesianas, denominadas *Hugin Lite*, UnBBayes e BNPC, para que uma delas fosse integrada a um ambiente de desenvolvimento, a fim de construir um protótipo com os seus recursos de aprendizagem a partir de bases de dados e, assim, facilitar a aquisição de conhecimento em sistemas especialistas probabilísticos. Após o estudo, foi escolhida a API do BNPC para realizar a integração com o ambiente de desenvolvimento *Visual Basic*, bem como foi construído o protótipo *VisionBayes* com uma interface gráfica intuitiva, que utilizou os recursos dessa API para aprendizagem automatizada de redes bayesianas. Nos testes realizados no *VisionBayes* com uma base de dados médica a respeito da prevalência do Diabetes Mellitus tipo dois em sete bairros da cidade de Criciúma, gerou-se uma rede bayesiana, que apresentou resultados adequados, refletindo o comportamento da população estudada.

Palavras-Chave: Descoberta de Conhecimento em Bases de Dados, Mineração de Dados, Redes Bayesianas, Aprendizagem Automatizada, BNPC.

ABSTRACT

The quantity of information stored in nowadays databases is so massive that it is difficult to analyze their data. Behind this huge amount of information there might be hidden significant relationships that will be found only by the use of suitable intelligent techniques. The process of Knowledge Discovery in Databases (KDD) groups together tasks and methods for the extraction of relevant knowledge from these databases, by means of its main part, the data mining, which can be done in several different ways, among them those based on Bayesian networks. In the realization of this research it was analyzed the Application Programming Interfaces (API) of the shells of data mining in Bayesian networks, known as Hugin Lite, UnBBayes e BNPC, so that one of them could be integrated to a development environment, with the purpose of building a prototype with its resources for learning from databases and, therefore, facilitating the knowledge acquisition in probabilistic expert systems. Completed the research, the API of the BNPC was chosen for the realization of an integration with the Visual Basic development environment, and the prototype VisionBayes was built with an intuitive graphical interface that used the resources of that API for the automated learning of Bayesian networks. In the tests performed in the VisionBayes with a medical database, in relation to the prevalence of diabetes mellitus type two in seven neighborhoods of Criciuma city, a Bayesian network was generated that presented adequate results, reflecting the behavior of the studied population.

Keywords: Knowledge Discovery in Databases, Data Mining, Bayesian Networks, Automated Learning, BNPC.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API	*Application Programming Interface*
BKD	*Bayesian Knowledge Discoverer*
BNPC	*Belief Network Power Constructor*
DAO	*Data Access Objects*
DCBD	Descoberta de Conhecimento em Bases de Dados
DLL	*Dinamic Linked Library*
GUI	*Graphical User Interface*
HDE	*Hugin Decision Engine*
ISAM	*Indexed Sequential Access Method*
KDD	*Knowledge Discovery in Database*
MSBN	*Microsoft Belief Network Tools*
ODBC	*Open Database Connectivity*
RB	Rede Bayesiana
SE	Sistema Especialista
SEP	Sistema Especialista Probabilístico

SUMÁRIO

1 INTRODUÇÃO

Em aplicações baseadas em conhecimento, como os sistemas especialistas probabilísticos, uma tarefa importante e muitas vezes difícil de realizar é a aquisição de conhecimento, podendo ser feita de duas maneiras: por meio de seu modelo clássico ou de forma automatizada.

Destaca-se no modelo clássico, a técnica de entrevista com o especialista da área de conhecimento, sendo esse modelo lento e trabalhoso na modelagem do conhecimento, pois na maioria dos casos, o especialista tem dificuldade de expressar e associar valores para as probabilidades condicionais que compõem a parte quantitativa, bem como estabelecer as relações que compõem a parte qualitativa de uma rede bayesiana. Considerando esse aspecto, o modelo automatizado, baseado na descoberta de conhecimento em bases de dados, prevê a participação do especialista na supervisão do processo de aquisição de conhecimento. Este modelo pode ser mais indicado nos dias atuais por oferecer um conjunto de técnicas que buscam e tratam a informação presente nessas bases.

A tecnologia de mineração de dados em redes bayesianas disponibiliza uma série de algoritmos que atuam em bases de dados para a aquisição de conhecimento automatizada, proporcionando, assim, vários benefícios, como por exemplo, o ganho de tempo nesse processo de aquisição e a visualização das informações por meio de uma estrutura gráfica – concebida pelas redes bayesianas – que provê a facilidade de comunicação entre o engenheiro do conhecimento e o especialista no desenvolvimento de um sistema especialista probabilístico.

Nesse sentido, esta pesquisa tende a continuar o Trabalho de Conclusão de Curso de Manarin (2004), que abordou a análise de *shells freewares* de mineração de

dados em redes bayesianas, e desenvolver um protótipo que utilize os recursos da API de uma dessas *shells* de extração de conhecimento, facilitando, dessa forma, a aquisição de conhecimento.

1.1 OBJETIVO GERAL

Desenvolver o protótipo de um sistema para aquisição de conhecimento em sistemas especialistas probabilísticos que integre um ambiente de desenvolvimento com a *Application Programming Interface* (API) de uma *shell* de mineração de dados em redes bayesianas.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são:

a) compreender o processo de descoberta de conhecimento em redes bayesianas;
b) analisar os recursos da API de uma *shell* de mineração de dados em redes bayesianas;
c) oferecer, via interface com o usuário, um módulo de aquisição de conhecimento para sistemas especialistas probabilísticos;
d) disponibilizar um mecanismo de aquisição de conhecimento em bases de dados para construção de redes bayesianas;
e) documentar o processo de integração do ambiente de desenvolvimento com uma *shell* de mineração de dados em redes bayesianas.

1.3 JUSTIFICATIVA

Nos bancos de dados atuais, a maioria das informações é encontrada de forma não-estruturada e em grande quantidade, tornando-se necessário o uso de algum tipo de processo que identifique e interprete padrões de comportamento, para que se possa extrair relações significativas, normalmente não observadas pelas formas de análise de dados tradicionais (CARVALHO, 2002). Partindo desse contexto, o processo que envolve a descoberta de conhecimento em bases de dados, por meio de sua etapa de mineração de dados, vem suprir essa necessidade, sendo responsável por extrair conhecimento de um conjunto de informações e propiciar uma visão mais organizada e analítica, contribuindo no processo de tomada de decisão.

A fase de mineração dados pode ser realizada de diversas maneiras, destacando-se a fundamentada por meio de redes bayesianas, que, a partir de bases de dados, constrói sua estrutura com as relações das informações encontradas nessas bases e seus valores de incidência.

Estudos nesta área têm sido realizados, tanto em nível acadêmico, com trabalhos de mestrado e doutorado, quanto em nível comercial, com a implementação de ferramentas, para a descoberta de conhecimento por meio de redes bayesianas.

Dessa forma, a construção do protótipo pela integração da API de uma *shell* de mineração de dados em redes bayesianas visa utilizar seus recursos de aprendizagem na aquisição de conhecimento automatizada, a partir de bases de dados, para sistemas especialistas probabilísticos.

1.4 ESTRUTURA DO TRABALHO

O trabalho está estruturado em 8 capítulos. O capítulo 1 apresenta a introdução com os objetivos e a justificativa a respeito da pesquisa.

O conceito de descoberta de conhecimento em bases de dados, retratando suas etapas, principalmente a de mineração de dados, é abordado no capítulo 2.

O capítulo 3 trata da questão da aquisição e representação do conhecimento em sistemas especialistas probabilísticos.

O capítulo 4 conceitua redes bayesianas, relatando suas topologias e como são realizadas as inferências.

Uma abordagem sobre mineração de dados em redes bayesianas e sua utilização para a aquisição de conhecimento é retratada no capítulo 5.

O capítulo 6 apresenta alguns trabalhos de pesquisas referentes ao processo de descoberta de conhecimento utilizando mineração de dados em redes bayesianas.

O trabalho desenvolvido, com a descrição de cada etapa da metodologia de desenvolvimento, bem como os resultados obtidos, é abordado no capítulo 7.

E por fim, tem-se a conclusão, com a sugestão de alguns trabalhos futuros.

2 DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS

O avanço tecnológico e o poder computacional proporcionaram, nas últimas décadas, um grande aumento da capacidade de processamento e armazenamento de dados oriundos dos mais diversos campos. Esta enorme quantidade de informações pode esconder detalhes e relações significativas que não seriam encontrados pelo homem em simples processos de análise, exigindo, desta forma, o auxílio de ferramentas inteligentes apropriadas (GOLDSCHIMIDT; PASSOS, 2005).

A descoberta de conhecimento em bases de dados (DCBD), conhecida originalmente por *Knowledge Discovery in Database* (KDD) é uma linha de pesquisa que vem se desenvolvendo e crescendo rapidamente, ligada ao benefício que esta pode trazer às organizações, satisfazendo necessidades práticas, sociais, econômicas, entre outras. O motivo para este crescimento é que o processo de KDD dispõe de uma tecnologia de coleta, armazenamento e gerenciamento de grande quantidade de dados, os quais possuem informações valiosas, como tendências e padrões, que, por meio de sua descoberta e extração automática, podem contribuir na tomada de decisão (REZENDE, 2003).

O termo KDD surgiu pela primeira vez em um *workshop* realizado em 1989, onde se enfatizou que o produto final do processo de descoberta em bases de dados era o conhecimento (FAYYAD et al, 1996 apud ROMÃO, 2002).

Em 1995, realizou-se a Primeira Conferência Internacional sobre KDD. No ano seguinte foi realizada a segunda, denominada KDD-96, evento que tem sido repetido anualmente, contando com os principais pesquisadores e importantes publicações que contribuem nesta área de pesquisa (ROMÃO, 2002).

O processo de KDD é formado das etapas descritas a seguir.

2.1 PROCESSO DE DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS

O processo de KDD se baseia em um fluxo de operações que vai desde a obtenção dos dados até a avaliação e interpretação do conhecimento adquirido, conforme ilustra a Figura 1:

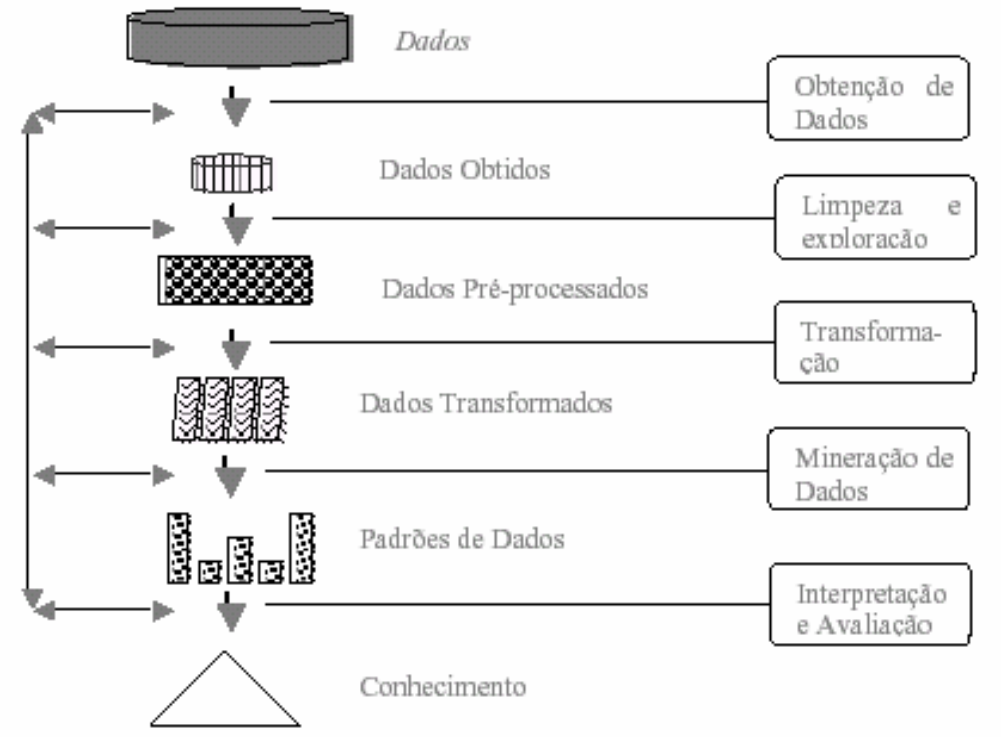

Figura 1. Fluxo básico do processo de KDD
Fonte: MELLO, L. (2001)

Este fluxo básico de operações apresenta características interativas e iterativas (MELLO, 2001). Segundo Goldschimidt e Passos (2005), o termo interativo mostra que é necessária a participação do homem, sendo ele o responsável pelo controle do processo de KDD, análise e interpretação de seus resultados. Já o termo iterativo, indica a possibilidade de repetições parciais ou totais do processo, na busca de melhores resultados.

A interatividade entre os envolvidos no processo de KDD como um todo e a iteratividade de seu fluxo são fundamentais para o sucesso na obtenção de conhecimento relevante (NETO; LIMA; AFONSO, 2002).

Segundo Rezende (2003), existem diversas abordagens sobre a real divisão das etapas que formam o KDD. Alguns autores o dividem em nove etapas distintas. Para melhor compreensão, o processo de KDD é tratado em apenas três grandes etapas operacionais, como mostra a Figura 2, que, de certa forma, resumem as nove mencionadas anteriormente: pré-processamento, extração de padrões (mineração de dados) e pós-processamento, descritas a seguir (GOLDSCHIMIDT; PASSOS, 2005), (ROMÃO, 2002):

a) pré-processamento: aplicação de uma série de funções relacionadas à captação, tratamento, integração, limpeza, redução do volume de dados e seleção de atributos principais;
b) mineração de dados: após os dados serem limpos, padronizados e organizados, esta etapa refere-se a escolha, configuração e aplicação de algum algoritmo de um método de inteligência artificial ou modelos estatísticos, para a extração de padrões embutidos nos dados analisados, resultando em conhecimento;
c) pós-processamento: contempla a avaliação, análise e seleção do conhecimento extraído na etapa anterior e sua possível utilização ou não.

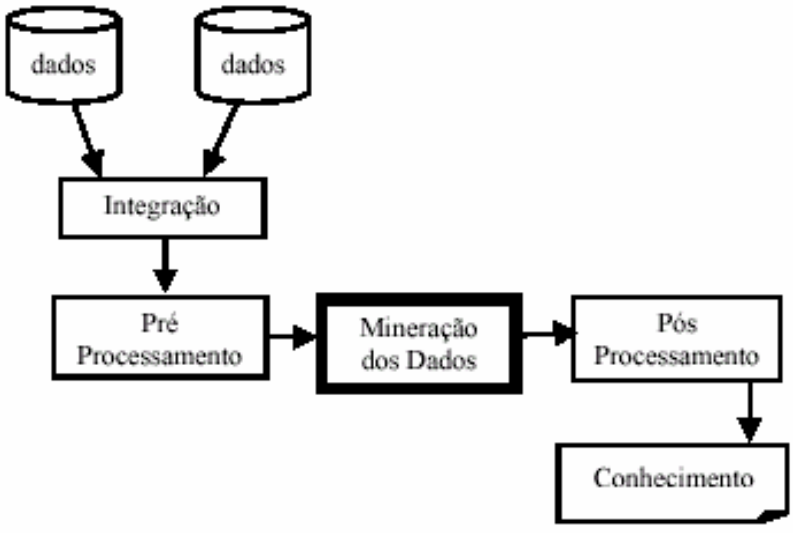

Figura 2. Etapas para a descoberta de conhecimento
Fonte: ROMÃO, W. (2002)

A etapa de mineração de dados é fundamental no processo de KDD e apresenta uma série de tarefas e métodos para a extração de conhecimento, como será visto a seguir.

2.2 MINERAÇÃO DE DADOS

Conforme Carvalho (2002, p. 7), mineração de dados é “o uso de técnicas automáticas de exploração de grandes quantidades de dados de forma a descobrir novos padrões e relações que, devido ao volume de dados, não seriam facilmente descobertos a olho nu pelo ser humano”.

Com certa freqüência, os termos KDD e mineração de dados são tratados como sinônimos. Mas, deve-se salientar que mineração de dados é uma das etapas deste processo, sendo considerada a principal, pois é a responsável pela busca e extração de conhecimentos novos e úteis por meio da aplicação de uma série de técnicas – representadas em algoritmos – sobre uma base de dados, podendo ser extremamente útil em relação a tomada de decisão nos mais diversos domínios de aplicação (CARVALHO, 2002), (GOLDSCHIMIDT; PASSOS, 2005).

Antes da escolha e aplicação do algoritmo de um método de mineração de dados, é necessário saber qual o tipo de conhecimento que se deseja extrair dos dados, e este está associado às tarefas de mineração de dados (REZENDE, 2003).

2.2.1 Tarefas de Mineração de Dados

A escolha de uma tarefa está associada ao tipo de conhecimento final, que será resultado do processo de mineração de dados.

Segundo Rezende (2003), as tarefas são divididas em atividades de predição e atividades de descrição.

As atividades preditivas (ou supervisionadas) se baseiam em um conjunto de amostras com classes conhecidas para identificar a classe de uma nova amostra de dados (MOTTA, 2005). As duas principais tarefas deste tipo, que visam à tomada de decisão, são a classificação e a regressão (REZENDE, 2003).

As atividades descritivas (ou não-supervisionadas) não possuem classes conhecidas, procurando identificar no conjunto de dados, padrões de comportamento comuns entre eles (MOTTA, 2005). Destacam-se as tarefas de *clusterização*, regras de associação e sumarização, as quais visam o suporte à decisão (REZENDE, 2003).

A seguir demonstram-se as principais tarefas de mineração de dados (GOLDSCHIMIDT; PASSOS, 2005):

a) **regras de associação:** esta tarefa busca por itens que ocorram de maneira simultânea em transações do banco de dados. Pode-se citar como exemplo, sua aplicação à área de *marketing* de uma grande rede de supermercados norte-americana, que constatou uma associação entre a compra de fraldas e a de cerveja, realizada principalmente por homens. Isto fez com que houvesse uma modificação das gôndolas do supermercado, aproximando estes dois produtos;

b) **classificação:** esta tarefa busca criar uma função que mapeie um conjunto de registros em um conjunto de classes, para que, caso apareça um novo registro, este possa ser enquadrado em alguma destas classes já definidas. Considerando uma empresa financeira que promove empréstimos a seus clientes, um exemplo de aplicação desta tarefa seria descobrir uma função que, com base em seu histórico de dados, os

dividissem em clientes que pagam em dia e clientes inadimplentes. Assim a empresa poderia prever, com base nesta função, o comportamento de seus novos clientes;

c) **regressão:** muito semelhante a tarefa de classificação, esta tarefa também busca descobrir uma função que mapeie os registros de uma base de dados, restringindo-se apenas a atributos numéricos. Como exemplo, pode-se citar: definição do limite do cartão de crédito de um determinado cliente ou a estimativa da probabilidade de vida de um paciente com base nos resultados de seus exames;

d) ***clusterização*:** esta tarefa prevê a separação dos dados que possuem um certo grau de afinidade ou propriedades similares em subconjuntos ou *clusters*, distinguindo-se uns dos outros. A *clusterização* se difere da tarefa de classificação, já que esta última apresenta classes definidas e a primeira precisa identificar grupos automaticamente. Como exemplo, uma empresa de telecomunicação pode usar este tipo de tarefa para obter grupos de clientes que utilizam o mesmo serviço oferecido por ela;

e) **sumarização:** procura, identifica e indica características semelhantes em um conjunto de dados. É comum sua utilização em cada um dos *clusters* obtidos pela *clusterização*. Um exemplo desta tarefa seria analisar os assinantes de uma revista semanal, buscando a partir dos dados de seus clientes, características semelhantes a uma boa parte deles, para que o departamento de *marketing* possa oferecer a revista a novos clientes com o mesmo perfil.

Aliadas as tarefas, que fornecem diversos modelos de padrões de dados, estão os métodos de inteligência artificial, que dispõem de uma série de algoritmos para mineração de dados.

2.2.2 Métodos de Mineração de Dados

Existem diversos tipos de métodos e, conseqüentemente algoritmos, para mineração de dados, destacando-se as redes neurais artificiais, a lógica nebulosa, os algoritmos genéticos, as árvores de decisão, os métodos estatísticos, entre outros.

As redes neurais artificiais se baseiam na construção de sistemas inteligentes por meio de modelos baseados nas conexões, estrutura e funcionamento do cérebro humano (BITTENCOURT, 2001). Elas têm capacidade de aprendizado, generalização, associação e abstração. Tendem a aprender por processos de repetição, ou seja, por experiência, melhorando gradativamente seu desempenho à medida que são treinadas e seus erros corrigidos (REZENDE, 2003), (AURÉLIO; VELLASCO; LOPES, 1999). O método de redes neurais artificiais é apropriado às tarefas de classificação, regressão e *clusterização* (GOLDSCHIMIDT; PASSOS, 2005).

A lógica nebulosa (ou lógica f*uzzy*), baseada na teoria matemática dos conjuntos e seus graus de pertinência a cada um deles, permite a modelagem do raciocínio em situações onde o conhecimento humano é impreciso (REZENDE, 2003). Em mineração de dados, as tarefas desempenhadas por este método são a classificação, a *clusterização* e a regressão (GOLDSCHIMIDT; PASSOS, 2005).

Os algoritmos genéticos são modelos de busca e otimização, baseados na teoria biológica da evolução natural de Darwin, e usam os mecanismos de seleção, mutação e reprodução para indicar as populações mais aptas a se adaptarem a algum

meio. Eles têm se mostrado um método eficaz no tratamento e resolução de problemas complexos de otimização, sendo responsáveis pela busca de soluções otimizadas (BARONE, 2003), (BARRETO, 2001). Em mineração de dados, os algoritmos genéticos são apropriados às tarefas de classificação, *clusterização* e sumarização (GOLDSCHIMIDT; PASSOS, 2005).

Uma árvore de decisão é um modelo de conhecimento, representado por um conjunto de regras, em que cada nó interno representa uma decisão sobre um atributo que determina como os dados estão divididos pelos seus nós filhos (folhas). As regras têm início na raiz da árvore e determinam o andamento pelas suas folhas (REZENDE, 2003), (GOLDSCHIMIDT; PASSOS, 2005). Este método está associado às tarefas de classificação e regressão (DIAS, 2001).

Os métodos estatísticos fornecem modelos e técnicas tradicionais que são utilizadas para análise e interpretação de dados. Estão associados às tarefas de classificação e *clusterização* (GOLDSCHIMIDT; PASSOS, 2005). Dentre os métodos estatísticos, destacam-se as redes bayesianas (RB), que fornecem uma estrutura intuitiva, formada por nós, representando as variáveis aleatórias (parte qualitativa da rede) e suas relações uns com os outros por meio de probabilidades (parte quantitativa) (KOEHLER; NASSAR, 2002).

3 AQUISIÇÃO DE CONHECIMENTO EM SISTEMAS ESPECIALISTAS PROBABABILÍSTICOS

A aquisição de conhecimento é uma tarefa fundamental na construção da base de conhecimento em sistemas especialistas probabilísticos (SEP). Pode ser definida como o processo de modelagem de problemas e soluções por meio de técnicas manuais ou automáticas, que visam extrair o conhecimento do especialista em um domínio específico, facilitando a criação deste tipo de sistema inteligente (REZENDE, 2003).

Em sua maneira clássica, a aquisição de conhecimento é realizada por meio de entrevistas, onde o especialista é o responsável por transmitir o conhecimento. Esse processo não é eficiente para um SEP, pois os especialistas não são muito claros em associar valores de probabilidade ao grau de solução dos problemas, o que acaba consumindo muito tempo, visto que, é natural do ser humano apresentar certa dificuldade em associar números ao quanto ele acredita em determinada situação. Sendo assim, tem-se buscado novas maneiras de extrair conhecimento válido por meio métodos de mineração de dados (KOEHLER; NASSAR, 2002), (NASSAR, 2005).

A seguir, serão vistas as características dos SEP e a forma como o conhecimento adquirido é representado em sua base de conhecimento.

3.1 SISTEMAS ESPECIALISTAS PROBABILÍSTICOS

Em alguns domínios de aplicação, o raciocínio para a solução de um determinado problema não é exato, sendo acompanhado por um certo grau de incerteza, como acontece em certos diagnósticos médicos (NASSAR, 2005). Para tratar desta questão do conhecimento não totalmente definido, ou seja, incerto, surgiram os SEP,

denominados assim, por apresentar a mesma arquitetura dos sistemas especialistas (SE) clássicos.

Os SEP são sistemas computacionais com conhecimento específico em um determinado domínio de aplicação, que devem se comportar de forma semelhante a um especialista humano na resolução de problemas (BARRETO, 2001).

De acordo com Barreto (2001), para a construção de um SEP é necessário:

a) uma fonte de conhecimento, que pode ser adquirida por um especialista ou de forma automatizada;
b) o conhecimento que foi obtido deve ser transformado de uma maneira conveniente e armazenado no computador;
c) este conhecimento compreende os fatos sobre o problema a ser resolvido e as regras que expressam o raciocínio de como chegar a solução do mesmo;
d) e ainda, freqüentemente, em casos onde o usuário final não concorde com a sugestão dada pelo SEP, é necessário dispor de um mecanismo que seja capaz de gerar explicações sobre como ele chegou a determinada conclusão.

A Figura 3 apresenta a arquitetura básica de um SEP, que segundo Barone (2003) é formada pelos seguintes componentes:

a) **base de conhecimento:** armazena todo o conhecimento de um especialista em um domínio de aplicação a ser utilizado pelo mecanismo de inferência;
b) **mecanismo de inferência:** responsável pelo raciocínio de um SEP, examinando e procurando respostas, na base de conhecimento, às

consultas do usuário, transferindo os fatos e regras para a memória de trabalho[1];

c) **aquisição de conhecimento:** responsável pela atualização da base de conhecimento, que pode ser realizada via especialista, com a inclusão, alteração ou exclusão do seu próprio conhecimento, ou técnicas de mineração de dados;

d) **módulo de explanação:** informa de que maneira é realizado o raciocínio de uma determinada ação pelo sistema, relatando de que forma tal conclusão foi conseguida;

e) **módulo de interface:** responsável pela interação entre o sistema e o usuário.

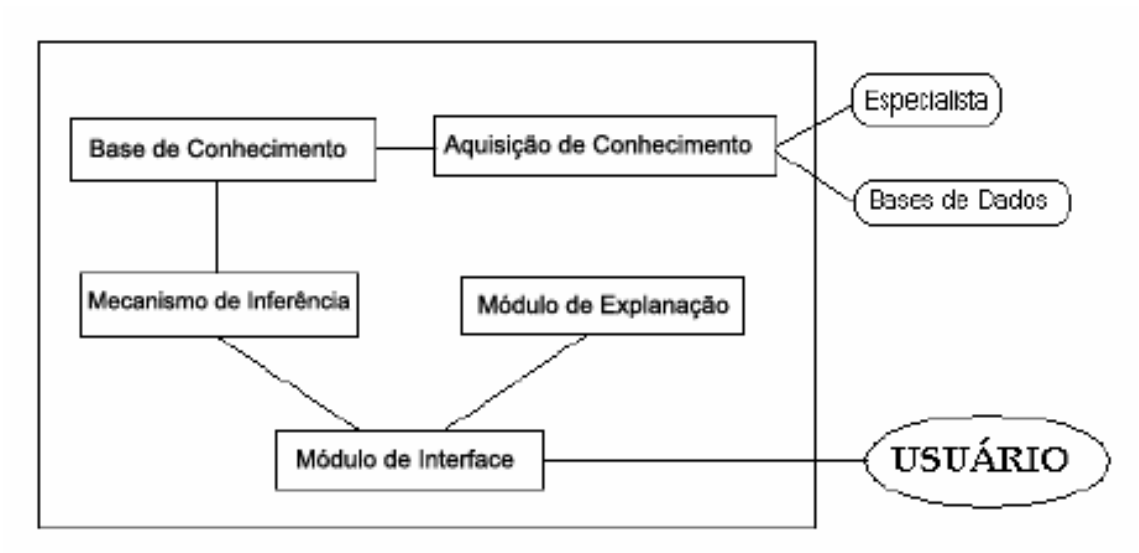

Figura 3. Arquitetura de um SEP
Fonte: Adaptado de NASSAR, S. (2005)

A diferença entre um SEP e um SE clássico reside na base de conhecimento formada pela distribuição das probabilidades condicionais.

Os SEP têm em sua base de conhecimento, fatos e regras, que representam o conhecimento de um especialista, aos quais são associadas as incertezas presentes no

[1] É o conjunto de informações fornecidas pelo usuário em tempo de execução.

domínio e as chances de sua ocorrência por meio de valores probabilísticos, já que o raciocínio do sistema deve considerar estes valores para que, a partir dos dados de entrada, sejam realizadas associações entre o vetor de probabilidades e o conjunto de hipóteses diagnósticas. A conclusão do sistema será a hipótese com maior probabilidade de ocorrência, visto que, a esta conclusão está associado o grau de certeza da resposta do sistema (NASSAR, 2005).

O conhecimento em algum domínio de aplicação para ser armazenado e, conseqüentemente utilizado por meio de inferências, deve, essencialmente, ser representado de alguma maneira (FERNANDES, 2003).

Em SEP, a representação do conhecimento do especialista, resultado de um processo de aquisição de conhecimento, é realizada por meio de uma RB, onde é possível visualizar suas probabilidades condicionais e realizar inferências com a mesma.

A representação do conhecimento incerto, típica dos SEP, é demonstrada a seguir.

3.1.1 A Incerteza

Os SEP são desenvolvidos para resolver uma variedade de problemas, e o conhecimento para chegar a solução destes não é bem definido. Muitos problemas do mundo real apresentam estas características de incerteza. Os especialistas humanos que trabalham nestes tipos de domínio têm condições de fornecer respostas corretas e tomar decisões quando a informação se mostra incerta, incompleta e até mesmo contraditória. Um SEP, para ser eficiente e confiável, deve se comportar da mesma forma como o especialista humano na resolução destes tipos de problemas (NASSAR, 2005).

Segundo Barreto (2003), os formalismos para tratar a incerteza na representação do conhecimento podem ter uma abordagem simbólica ou numérica.

A simbólica é adequada para tratar a informação incompleta (algumas respostas são conhecidas), mas não sendo eficiente para tratar a imprecisa (as respostas são conhecidas, mas são aproximadas), pois não se consegue quantificar este tipo de incerteza (BARONE, 2003).

Nassar (2005) completa dizendo que o método simbólico trata as incertezas por meio de regras de inferência, que representam as exceções no raciocínio do especialista, sendo viável esta abordagem quando se trabalha com uma pequena quantidade de exceções. Quando estas são muitas, torná-las explícitas pode ser muito difícil, inviabilizando esta forma de representação. Deve-se considerar também que as exceções devem ser identificadas e representadas no SEP antes da realização de inferências e combinações dos dados de entrada.

Já a abordagem numérica propaga a incerteza numericamente por meio das inferências e combinações de evidências, sendo que as principais formas de representação são por meio de (NASSAR, 2005), (RUSSEL; NORVIG, 2004):

a) **fatores de certeza:** utilizado nos primeiros SE clássicos, este método associa uma informação a um determinado grau de certeza;
b) **teoria dos conjuntos difusos:** representa a incerteza por imprecisão, ou seja, em casos onde os conceitos são vagos e os limites são imprecisos. Método utilizado para representar termos lingüísticos, como por exemplo, alto, baixo, velho, jovem, quente, frio, muitos, poucos. Na teoria dos conjuntos clássicos, um elemento pertence a um determinado conjunto. Já na teoria dos conjuntos difusos, um mesmo elemento pode

pertencer a mais de um conjunto ao mesmo tempo por meio de seu grau de pertinência a determinado conjunto, que varia de 0 a 1;

c) **teoria da probabilidade:** representa a incerteza por aleatoriedade, ou seja, não há como prever com certeza como um novo caso se comportará, mesmo diante do conhecimento de casos anteriores do domínio de aplicação. Utiliza uma estrutura de eventos aleatórios, onde a probabilidade destes eventos ocorrerem está entre os valores 0 e 1. As probabilidades assumem a forma A→B, onde a probabilidade de um evento B (conseqüência) ocorrer está associada a ocorrência de um outro evento A (causa) e são calculadas seguindo o teorema de Bayes, descrito no item 4.2;

d) **teoria da evidência:** conhecida como a teoria de Dempster-Schafer. Semelhante ao teorema de Bayes, este tipo de representação é feita por meio de medidas de crença, com valores de intervalo, baseada no cálculo da probabilidade de que a evidência admita uma proposição, ao invés de calcular a probabilidade dela.

Segundo Barreto (2001), o raciocínio probabilístico, baseado na teoria da probabilidade descrita anteriormente, é talvez a maneira mais antiga para tratar os mecanismos de incerteza, pois apóia-se em informações probabilísticas sobre os fatos do domínio, para chegar a conclusão de um novo fato, associando a esta, uma probabilidade.

A seguir, serão vistas as RB, estruturas fundamentadas na teoria da probabilidade, que representam o conhecimento incerto e constituem a base de conhecimento em SEP.

4 REDES BAYESIANAS

Uma RB é uma estrutura de representação do conhecimento, por meio de um modelo probabilístico, que expressa as relações de causa/conseqüência e valores de probabilidade entre as variáveis de um domínio de conhecimento (NETO; LIMA; AFONSO, 2002).

Pode ser considerada um grafo acíclico orientado onde os nós representam um conjunto de variáveis aleatórias e, a ligação das setas ou vínculos orientados unindo os nós, representa a dependência probabilística entre as variáveis associadas. Cada nó possui uma distribuição de probabilidade condicional dos valores que podem ser assumidos pela variável aleatória associada ao nó, dado os valores de seus nós pais (BARONE, 2003), (RUSSEL; NORVIG, 2004).

A Figura 4 mostra um exemplo de uma RB, formada de três variáveis (nós) e duas relações. O nó pai Gripe possui dois nós filhos, Febre e Tosse, os quais demonstram ser dependentes e influenciados pelo nó pai.

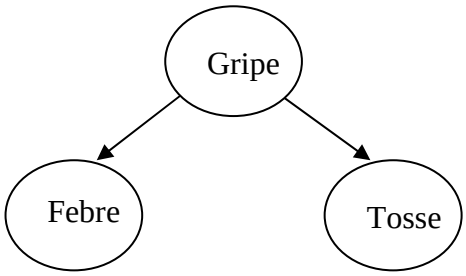

Figura 4. Exemplo de uma RB

As RB são compostas por uma parte qualitativa que representa as variáveis e suas relações de dependência e por uma quantitativa, representada pela distribuição das probabilidades. Nestas redes, pode-se calcular a probabilidade de um evento acontecer, mediante a ocorrência de outro, e este cálculo é realizado pelo teorema de Bayes, descrito no item 4.2 (KOEHLER; NASSAR, 2002).

Segundo Barone (2003), o uso de redes bayesianas apresenta as seguintes vantagens:

a) representa e manipula a incerteza com base em princípios matemáticos fundamentados;
b) modela o conhecimento do especialista de maneira intuitiva;
c) formalismo de representação do conhecimento que permite realizar qualquer um dos tipos de inferência probabilística, isto é, causal (das causas aos efeitos), diagnóstica (dos efeitos às causas), intercausal (entre as possíveis causas para um mesmo efeito) ou mista (combinação das anteriores).

A principal característica de uma RB é a habilidade de explorar a sua estrutura e reduzir o cálculo (da probabilidade condicional de um evento, dada a evidência) a uma série de procedimentos locais, usando somente os nós necessários, evitando calcular a função de distribuição de probabilidades conjuntas global, ou seja, de todos os nós (BARONE, 2003).

As partes quantitativa e qualitativa, que formam uma RB, interagem entre si, relacionando suas variáveis e seus valores condicionais e incondicionais probabilísticos, por meio da sua topologia.

4.1 TOPOLOGIA DAS REDES BAYESIANAS

A topologia da rede, formada pelo conjunto de variáveis e suas ligações umas com as outras, especifica os relacionamentos de independência condicional que são válidos no domínio de conhecimento. Em geral, o significado de uma seta entre uma variável X e outra Y, em uma rede construída corretamente, indica que existe uma

influência direta entre as duas. Após definir a topologia da RB, é necessário especificar uma distribuição de probabilidade condicional para cada variável da rede, dados seus pais (RUSSEL; NORVIG, 2004).

Existem três formatos de conexão entre as variáveis em uma RB, que determinam como a evidência se propaga (LADEIRA; VICARI; COELHO, 1999).

Na conexão serial, representada na Figura 5 (a), uma evidência em A influencia a crença em B, que influencia a crença em C, o mesmo acontecendo se começasse em C e fosse até A. Se B estiver instanciado e não for conhecido, haverá um bloqueio no caminho e este tipo de conexão não acontece. Na conexão divergente, Figura 5 (b), uma evidência em um ascendente de A influencia a crença sobre os seus filhos, exceto se A é instanciado, bloqueando o canal de comunicação com seus filhos. Na conexão convergente, Figura 5 (c), uma evidência em A ou em um dos seus descendentes influencia a crença nos pais de A. Caso A não for conhecido, então seus pais são independentes (LADEIRA; VICARI; COELHO, 1999).

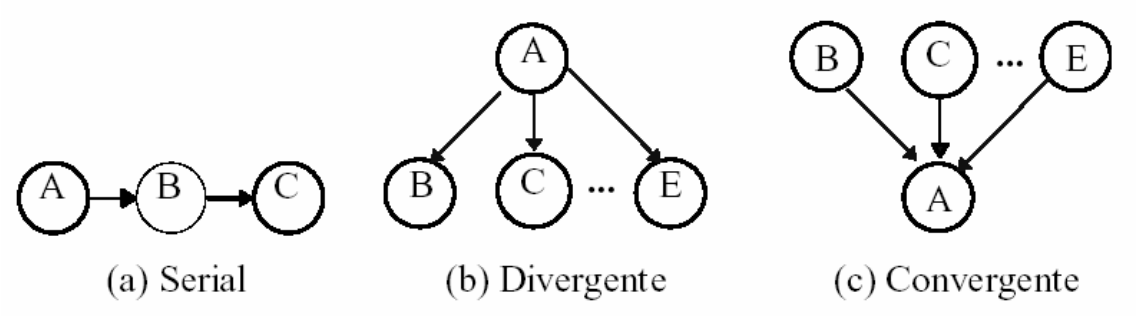

Figura 5. Topologia das RB
Fonte: LADEIRA, M.; COELHO, W.; VICARI, R. (1999)

Vistas as topologias, será descrita a seguir, a forma como se dá o processamento das probabilidades em uma RB por meio das inferências.

4.2 INFERÊNCIA BAYESIANA

O raciocínio em um SEP é baseado na realização de inferências probabilísticas (cálculo da probabilidade de um evento, dada todas as evidências disponíveis). Este cálculo dos valores de probabilidade é estimado por meio da utilização da teoria da probabilidade, mais precisamente pelo teorema de Bayes, cuja fórmula é mostrada abaixo (LADEIRA; VICARI; COELHO, 1999), (NASSAR, 2005).

$$P(B \mid A) = \frac{P(A \mid B)\, P(B)}{P(A)}$$, onde:

P (B | A) = probabilidade da hipótese acontecer, se a evidência aconteceu.

P (A | B) = probabilidade de acontecer a evidência, se a hipótese for verdadeira.

P (B) = probabilidade do acontecimento da hipótese (probabilidade *a priori*)

P (A) = probabilidade de ocorrência da evidência em todas as hipóteses.

No cálculo da probabilidade de um certo evento acontecer, todas as declarações de probabilidade devem apontar a evidência, concordando com a probabilidade que está sendo avaliada. Sempre que existirem novas informações, as avaliações de probabilidade devem ser atualizadas para refletir novas evidências (RUSSELL; NORVIG, 2004).

Na utilização do teorema de Bayes é necessário uma probabilidade condicional e duas probabilidades incondicionais conhecidas para se calcular uma nova probabilidade condicional, até então desconhecida.

Segundo Luger (2004), uma probabilidade incondicional ou *a priori*, é a probabilidade atribuída a um evento, na falta de conhecimento que suporte sua ocorrência ou ausência, ou seja, antes de qualquer evidência. Já uma probabilidade condicional ou *a posteriori*, é a probabilidade do acontecimento de algum evento, dado que exista uma evidência.

Por exemplo, a probabilidade de se ter infarto agudo do miocárdio (P(infarto)) caso seja sedentário (P(sedentário)) é calculada, de acordo com as suas probabilidades incondicionais, ou seja. 11,2 e 39,3, respectivamente, obtendo-se como resultado, 14,1% de chances, sem nenhuma outra evidência a ser analisada.

Visto como acontece o processo de consultas em SEP utilizando RB como forma de representação do conhecimento, a seguir mostrar-se-á como utilizar este método para extração de conhecimento em bases de dados.

5 MINERAÇÃO DE DADOS EM REDES BAYESIANAS

Mineração de dados e o processo de KDD baseados na tecnologia de RB são uma das áreas de pesquisa da inteligência artificial mais requisitadas nas últimas décadas, com um grande número de aplicações. As RB oferecem uma estrutura unificada e intuitiva, onde é possível comparar diferentes hipóteses, de acordo com os nós da rede, tornando-as um dos melhores métodos analíticos para a tomada de decisão (KOEHLER; NASSAR, 2002).

A aquisição de conhecimento para a construção da base de conhecimento em SEP por meio de especialistas, geralmente, é um processo demorado, pois eles, muitas vezes não associam suas hipóteses a números, não conseguindo definir a estrutura da rede nem mesmo as suas probabilidades. Dessa forma, visando diminuir esse tempo gasto, a construção de RB de forma automatizada por meio da mineração de dados é capaz de estimar os valores das probabilidades e também identificar os nós da rede a partir de bases de dados, tornando o processo de aquisição mais rápido e possivelmente mais eficiente (KOEHLER; NASSAR, 2002), (CARNEIRO, 1999).

Sendo assim, a mineração de dados em RB deve ser realizada de acordo com as etapas que compreendem o processo de KDD.

5.1 KDD APLICADO À MINERAÇÃO DE DADOS EM REDES BAYESIANAS

O processo de KDD em RB utiliza a mesma metodologia do KDD, sendo composto também de três etapas responsáveis pelo pré-processamento, a mineração de dados propriamente dita e o pós-processamento.

Após a escolha de uma base de dados em um domínio de aplicação, o processo de KDD inicia-se com a etapa do pré-processamento, onde são aplicados alguns procedimentos, entre eles, tratamento de valores ausentes, repetidos, faltantes; redução do volume de dados; seleção de atributos principais entre os existentes na base; limpeza e integração (GOLDSCHIMIDT; PASSOS, 2005), (ROMÃO, 2002).

A etapa seguinte, chamada de mineração de dados é a principal do KDD, e em RB, utiliza algum mecanismo para a construção automatizada da RB pela aprendizagem de suas estruturas e probabilidades, conforme mostrará o item 5.2, a partir da base de dados tratada na etapa anterior.

Por fim, a etapa do pós-processamento, segundo Passos e Goldschimidt (2005), contempla a avaliação, análise e seleção do conhecimento extraído. Dessa forma, a RB gerada na fase de mineração de dados, pode ser analisada pelo especialista e, caso a aquisição de conhecimento tenha sido realizada de maneira correta, a RB poderá ser utilizada na construção de um SEP.

A Figura 6 resume de maneira ilustrativa o processo de KDD aplicado a RB.

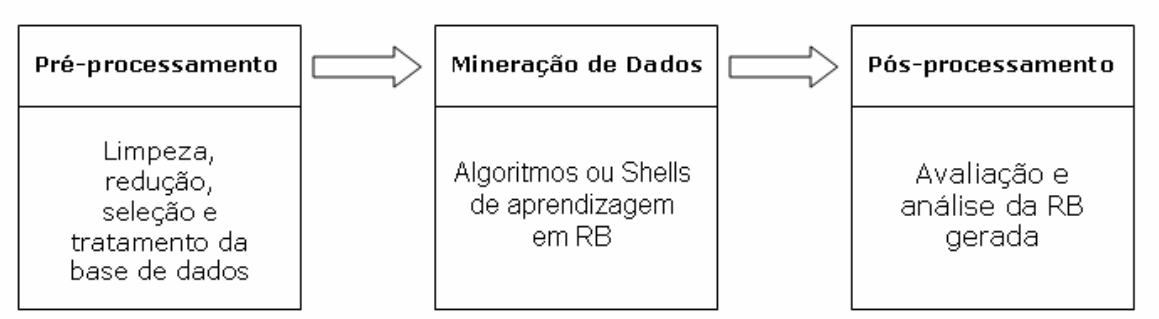

Figura 6. Processo de KDD para mineração de dados em RB

A seguir será visto como é realizada a aprendizagem em RB e as categorias de algoritmos utilizadas.

5.2 APRENDIZAGEM EM REDES BAYESIANAS

Aprendizagem, no contexto de RB, possui como entrada, um conjunto de dados e informação *a priori* e, como saída, uma RB (CARNEIRO, 1999).

Ainda segundo Carneiro (1999), a aprendizagem de RB envolve o processo de construção de uma rede probabilística que represente uma distribuição de probabilidades, em um determinado domínio de aplicação, sendo que a aprendizagem pode ser:

a) supervisionada: quando há a participação do especialista em quase todas as etapas de construção da RB, supervisionando-a com base na sua própria experiência;
b) não-supervisionada: forma de aprendizagem automatizada, dividida em duas partes distintas – a aprendizagem da estrutura e a dos parâmetros numéricos –, onde a construção da RB se dá automaticamente a partir da base de dados e tem a participação do especialista nas fases de coleta de dados e de teste.

Segundo Koehler (2002), a primeira etapa no processo de aprendizagem é gerar a RB que representa as relações entre as variáveis de um problema. Em seguida, deve-se especificar como a distribuição de probabilidades de cada nó será representada e, por fim, realiza-se uma estimativa das probabilidades condicionais com a base de dados, sendo importante observar, quais valores de variáveis devem ser considerados ou desconsiderados.

Os algoritmos de aprendizagem em RB possuem a capacidade de extrair de maneira automatizada, a partir de grandes bases de dados, a estrutura de uma RB, auxiliando o engenheiro do conhecimento e o especialista no processo de aquisição de

conhecimento para a construção de um SEP. Esses algoritmos estão divididos em duas categorias: baseados no método de busca e pontuação e na análise de dependência (KOEHLER; NASSAR, 2002).

5.2.1 Métodos de Busca e Pontuação

Nesta categoria, os algoritmos aprendem pela busca da estrutura que melhor se encaixa com os dados. Iniciam com um grafo sem arcos e usam algum método de busca para adicionar um arco ao grafo. Depois, utilizam-se de um método de pontuação para identificar se a nova estrutura é melhor do que a antiga. Se esta nova estrutura for melhor do que a anterior, mantém o novo arco adicionado e tentam adicionar outro. Este processo irá se repetir até que nenhuma estrutura nova seja melhor do que a estrutura atual (CHENG; BELL; LIU, 1998, tradução nossa).

Esses algoritmos, por serem na natureza heurística, não garantem encontrar uma boa solução e trabalham com um espaço de modelos de probabilidades maiores que o enfoque de análise de dependências (CARNEIRO, 1999).

Pode-se citar os seguintes algoritmos como pertencentes a esta categoria: árvores de Chow Liu, poliárvores de Rebane-Pearl, Kutató, HGC, Lam-Bacchus, CB, Benedict, Suzuki, Friedman-Goldszmidt, K2, EM e *bound and collapse* (CARNEIRO, 1999), (MANARIN, 2004).

5.2.2 Métodos Baseados em Análise de Dependências

Nesta categoria é assumido que a estrutura da rede representa perfeitamente as dependências e independências no domínio, ou seja, uma afirmação de independência

é representada por uma estrutura, se e somente se, ela é uma independência válida para o domínio. A validade de uma independência pode ser verificada realizando-se um teste estatístico usando um banco de dados sobre o domínio. A diferença básica entre os vários algoritmos desta categoria está no modo como os conjuntos de variáveis são encontrados e nas regras de associação das direcionalidades (ACID; CAMPOS 1996, tradução nossa).

Esses algoritmos constroem RB analisando as relações de dependência entre os nós (variáveis) e, esses relacionamentos de dependência, são medidos utilizando algum tipo de teste de independência condicional, o qual pode não ser confiável em grandes conjuntos de condições a menos que o volume de dados seja grande (CARNEIRO, 1999), (KOEHLER et al, 2004).

Pode-se citar os seguintes algoritmos como pertencentes a esta categoria: Wermuth-Lauritzen, *Boundary* DAG, SRA, SGS, PC, NPC e CBL (CARNEIRO, 1999).

Essas duas categorias de algoritmos estão implementadas nas *shells* que realizam aprendizagem em RB, as quais podem ser utilizadas no processo de mineração de dados.

5.3 *SHELLS* DE APRENDIZAGEM EM REDES BAYESIANAS

A seguir, são mostradas algumas *shells* que realizam aprendizagem em RB e podem auxiliar no processo de descoberta de conhecimento em bases de dados.

5.3.1 *Belief Network Power Constructor*

O *Belief Network Power Constructor* (BNPC) é uma *shell* que recebe como entrada uma base de dados e, a partir dela, gera uma RB, incluindo sua estrutura e as suas respectivas probabilidades condicionais (CHENG, 2006).

Baseia-se no algoritmo CBL, que de uma forma geral, constrói redes analisando relacionamentos de independência condicional entre os nós (CHENG; BELL; LIU, 1998).

O BNPC é composto de um assistente gráfico com algumas etapas que auxiliam o usuário durante o processo de construção da RB. De forma geral, a cada passo o usuário vai definindo uma série de configurações a respeito da base de dados escolhida, podendo ser adotadas também as definições padrões. Como resultado final, o BNPC gera a RB que, por meio de seu editor gráfico, pode ser ajustada e visualizada apenas com sua parte qualitativa, pois a *shell* não realiza inferências.

O BNPC suporta diferentes tipos de bases de dados, desde arquivos texto, planilhas eletrônicas, até acessos remotos com bancos de dados via *Open Database Connectivity* (ODBC). Ele disponibiliza uma *Dinamic Linked Library* (DLL), com recursos de aprendizagem, que possibilita sua utilização a outras *shells* de RB, mineração de dados e também sua integração em ambientes de desenvolvimento (CHENG, 2006).

A *shell* encontra-se na versão 2.2 e pode ser adquirida gratuitamente no *site* do fabricante em http://www.cs.ualberta.ca/~jcheng/bnpc.htm.

5.3.2 *Bayesian Knowledge Discoverer*

O *Bayesian Knowledge Discoverer* (BKD) é uma *shell* que permite a construção de RB, a propagação de evidências e a aprendizagem a partir de bases de dados.

Baseia-se no algoritmo K2 para a aprendizagem e utiliza o método *bound and collapse* para estimar as probabilidades condicionais quando existem dados com valores ausentes (BAYESWARE, 2005), (CARNEIRO, 1999).

O BKD possui uma interface constituída de duas partes principais que interagem entre si: a base de dados e a rede. É organizada de uma forma que permite que a edição da base seja dentro de uma janela, sendo que o usuário pode também trabalhar e gerar outras RB em várias janelas da rede. A *shell* é composta de três componentes principais: *database browser*, que faz o acesso e a manipulação da base de dados; *database windows*, que permite ao usuário a manipulação da base de dados; e *network windows*, que contem a RB.

A *shell* possui recursos tridimensionais, facilitando a visualização das partes quantitativa e qualitativa da RB e dispõe de assistentes que auxiliam o usuário na aprendizagem a partir de bases de dados e na inferência das redes criadas (BAYESWARE, 2005).

O BKD suporta bases de dados nos formatos texto (.txt, .prn, .csv) e *bayesware discoverer databases* (.hdd), que é o padrão utilizado pela *shell*, e não disponibiliza DLL (BAYESWARE, 2005).

Possui as versões profissional, empresarial, acadêmica e a estudante, sendo que esta última é grátis, e pode ser adquirida em http://www.bayesware.com.

5.3.3 *Hugin*

O *Hugin* é uma s*hell* de origem dinamarquesa para construção de sistemas de apoio a decisão, com base na utilização de RB e diagramas de influência, que podem ser construídos manualmente, ou a partir de bases de dados.

Baseia-se nos algoritmos PC e NPC para a aprendizagem da estrutura e o algoritmo EM para estimar o valor das probabilidades condicionais em bases de dados que possuam campos com valores ausentes (HUGIN, 2005).

A *shell* é composto de um *Hugin Graphical User Interface* (GUI), de um motor de inferências e do *Hugin* API.

O *Hugin* GUI possibilita a criação e a alteração no modelo da RB, bem como realizar operações sobre as evidências e exibir a distribuição de probabilidades (HUGIN, 2005).

O motor de inferências que é utilizado pelo *Hugin* interface de usuário gráfica possui um compilador, responsável pela realização de consultas, e pode ser utilizado por meio das API disponíveis pela *shell* (HUGIN, 2005).

O *Hugin* API fornece uma série de API para diversas linguagens de programação, como C, C++, Java e *Visual Basic*, permitindo que programadores possam construir aplicações baseadas em RB e dispor dos recursos do motor de inferências da *shell* (HUGIN, 2005).

Possui versões comerciais, como a *developer* e a *explorer*, acadêmicas, como a *researcher*, *educational* e a *classroom* e uma versão gratuita, a *lite*.

A versão *lite* suporta bases de dados no formato texto (.dat), sendo limitada a 500 registros e pode ser adquirida em http://www.hugin.com.

5.3.4 *Genie*

O *Genie* é uma *shell* para construção de modelos de apoio a decisão, baseados em RB e diagramas de influência, de qualquer tamanho e complexidade, a partir de bases de dados ou manualmente pelo usuário (GENIE, 2006).

A *shell* é composta de uma interface gráfica, que permite a criação de vários modelos de RB ao mesmo tempo, sendo prática a alternância entre elas, a visualização das redes criadas, bem como a realização de inferências.

O *Genie* é acompanhado de uma API chamada *Smile*, que é uma biblioteca de funções onde estão implementados modelos probabilísticos que permitem criar, editar, salvar modelos gráficos, e assim, usá-los para o raciocínio probabilístico e a tomada de decisão sob a incerteza (GENIE, 2006).

Suporta bases de dados nos formatos texto (.txt, .dat, .csv) e *genie data files* (.gdat), que é o padrão utilizado pela *shell*, bem como acesso via ODBC.

O *Genie* foi desenvolvido pela Universidade de Pittsburgh, é gratuito e pode ser adquirido em http://genie.sis.pitt.edu.

5.3.5 UnBBayes

O UnBBayes é um projeto brasileiro da Universidade de Brasília que permite a construção de RB pelo usuário ou a partir de bases de dados.

Baseia-se nos algoritmos de busca e pontuação, K2 e B, e no de análise de dependência, CBL versões A e B (implementado na *shell* BNPC) que o usuário pode escolher para a realização da aprendizagem.

O UnBBayes é composto de um ambiente visual e interativo, e de uma API, ambos escritos em Java, para edição e compilação de RB, entrada e propagação de evidências, realização de inferências probabilísticas e para aprendizagem da topologia e/ou parâmetros da rede (LADEIRA et al, 2006).

Pela *shell*, para realizar aprendizagem em bases de dados, é necessário abrir um arquivo .txt, pois a *shell* suporta bases de dados somente no formato texto, e a seguir um assistente guia o usuário durante o processo. É possível escolher as variáveis e o algoritmo e, a seguir, definir as relações entre cada variável. Por fim, a RB é gerada, podendo ser editada e visualizada (estrutura e probabilidades) no próprio ambiente gráfico.

A *shell* é um *software* livre, gratuito, para uso não comercial, podendo ser adquirida em http://unbbayes.sourceforge.net.

Outras *shells* de mineração de dados em RB podem ser encontradas em http://directory.google.com/Top/Computers/Artificial_Intelligence/Belief_Networks/Software ou http://www.cs.ubc.ca/~murphyk/Bayes/bnsoft.html

O processo de KDD associado as RB tem gerado pesquisas relacionadas à extração de conhecimento a partir de bases de dados, o que pode facilitar a aquisição de conhecimento na construção de SEP. Alguns trabalhos sobre aprendizagem de RB serão vistos a seguir.

6 ESTADO DA ARTE

A seguir, são apresentados alguns trabalhos de pesquisa relacionando o processo de KDD em RB, pela utilização dos algoritmos de aprendizagem ou por meio das *shells*, já que na pesquisa realizada não foram encontrados trabalhos que realizaram sua integração via API.

6.1 AQUISIÇÃO DE CONHECIMENTO EM SISTEMAS ESPECIALISTAS PROBABILÍSTICOS POR MEIO DA DESCOBERTA DO CONHECIMENTO EM BASE DE DADOS PARA CONSTRUÇÃO DE REDES BAYESIANAS

Trabalho de conclusão de curso realizado na Universidade do Extremo Sul Catarinense. O trabalho consistiu em realizar o processo de aquisição de conhecimento em SEP, por meio do KDD, gerando RB. Realizou-se uma pesquisa das principais *shells freewares* de mineração de dados em RB, bem como uma comparação entre elas, sendo feita a escolha da *shell* BNPC para a aquisição de conhecimento, a partir de uma base de dados sobre Diabetes Mellitus tipo dois em sete bairros da cidade de Criciúma, resultando em uma RB, a qual foi analisada por um especialista neste domínio de aplicação e apresentou resultados adequados, refletindo o comportamento da população estudada (MANARIN, 2004).

6.2 APLICAÇÃO DE REDES BAYESIANAS PARA EXTRAÇÃO DE CONHECIMENTO DE BASES DE DADOS

Trabalho de conclusão de curso realizado na Universidade da Amazônia. O objetivo do trabalho foi compreender o processo de KDD, detalhando suas etapas, principalmente a de mineração de dados, analisar a aplicação de RB como forma de representação do conhecimento a partir de bases de dados, como também utilizar algumas *shells* para esta finalidade. Foi apresentado ao final do trabalho, como resultados conquistados, um estudo de caso sobre o risco operacional na atividade bancária, analisando-se uma base de dados de operações de crédito por meio das *shells* de mineração de dados BNPC, BKD e *Hugin Lite* (NETO; LIMA; AFONSO, 2002).

6.3 SEPE: SISTEMA ESPECIALISTA PROBABILÍSTICO PARA APOIO AO DIAGNÓSTICO DE POTENCIAL ECONÔMICO

Dissertação de mestrado realizada na Universidade Federal de Santa Catarina. O objetivo do trabalho foi desenvolver um sistema para apoiar o planejamento mercadológico a partir do conhecimento do potencial econômico dos municípios de Santa Catarina. Em sua elaboração, trabalhou-se com dados incertos sobre o mercado, as decisões são tomadas sem ter as informações necessárias e, para que o diagnóstico seja realizado de uma forma mais eficiente, os técnicos devem visualizar os dados relacionados ao potencial de uma forma gráfica para que sejam investigadas relações entre variáveis endógenas[2] e exógenas[3] aos Correios, por meio de evidências

[2] Variável exógena é a variável que não é determinada pelo modelo, são insumos introduzidos no modelo.
[3]Variável endógena é aquela determinada pelo modelo, cuja finalidade é explicar como variáveis exógenas influenciam as variáveis endógenas.

observadas. O SEPE possui uma base de conhecimento atualizável e empregou métodos de mineração de dados no processo de aquisição e atualização de conhecimento de uma base de dados (PASINI, 2002).

6.4 E-BAYES: SISTEMA DE AVALIAÇÃO DA EVASÃO ESCOLAR

Dissertação de mestrado realizada na Universidade Federal de Santa Catarina. O sistema utiliza uma RB e mineração de dados para apoiar o gestor de uma universidade a avaliar a evasão escolar de um curso universitário, permitindo fazer a previsão da permanência ou não de um aluno no curso a partir de suas características sócio-demográficas, buscando na base de dados, quais os alunos daquele curso são similares as características do aluno em questão (ROVARIS NETO, 2002).

6.5 APRENDIZADO AUTOMÁTICO EM REDES BAYESIANAS

Dissertação de mestrado realizada na Universidade de Brasília. O objetivo principal do trabalho foi mostrar o estado da arte da aprendizagem em RB, estudando seus algoritmos e comparando o desempenho, via *shells*, dos dois mais significativos ao processo de aprendizagem, o algoritmo *bound and collapse* pertencente à categoria dos métodos baseados em busca e pontuação e o algoritmo CBL utilizando o método de análise de dependência, ambos aplicados a uma base de casos comum, sendo que o último, mostrou-se mais eficiente. Outro algoritmo, o K2, pertencente à categoria dos métodos de busca e pontuação, foi implementado e aplicado à mesma base citada, e também mostrou-se eficiente, gerando um RB próxima da ideal (CARNEIRO, 1999).

6.6 MODELAGEM DE REDES BAYESIANAS A PARTIR DA IDENTIFICAÇÃO DE PADRÕES EM BASES DE DADOS

Tese de doutorado realizada na Universidade Federal do Rio Grande do Sul. O objetivo do trabalho foi realizar a análise dos algoritmos de aprendizagem em RB, de forma a extrair modelos de conhecimento a partir de bases de dados, fazendo com que estas novas informações possam auxiliar o especialista e o engenheiro do conhecimento na etapa de aquisição do conhecimento que compõe o processo de construção de um SE. Obteve-se como resultados que dois algoritmos são considerados os mais importantes, o CBL, que utiliza a análise de dependência entre as variáveis, e o K2, que utiliza o método de busca e pontuação, para gerar RB a partir de bases de dados, sendo que, o CBL está condicionado a um conhecimento do domínio de aplicação para ser gerada a rede (KOEHLER, 2002).

No próximo capítulo é descrita a metodologia utilizada na pesquisa para o desenvolvimento do protótipo.

7 MINERAÇÃO DE DADOS EM REDES BAYESIANAS UTILIZANDO A API DA *SHELL BELIEF NETWORK POWER CONSTRUCTOR* (BNPC)

O presente trabalho consiste na utilização dos recursos da API de uma *shell freeware* de mineração de dados em RB e um ambiente de desenvolvimento, para a aquisição de conhecimento automatizada em SEP, a partir de bases de dados.

A metodologia para a realização do trabalho foi iniciada com o levantamento bibliográfico a respeito dos temas da pesquisa. Em seguida, para o cumprimento dos objetivos, foram realizadas as seguintes etapas:

a) levantamento das *shells* de mineração de dados em RB que disponibilizam API;
b) seleção de uma *shell* e de um ambiente de desenvolvimento para a construção do protótipo;
c) estudo dos recursos da API da *shell* selecionada;
d) integração da API com o ambiente de desenvolvimento e construção de uma interface gráfica;
e) definição de uma base de dados para realização de testes.

*7.1 SHELLS D*E MINERAÇÃO DE DADOS EM RB QUE DISPONIBILIZAM API

Durante o processo de levantamento das *shells* de mineração de dados em RB, verificou-se que muitas delas não disponibilizam recursos de API e, das que disponibilizam, destaca-se que a maioria não possibilita a aprendizagem a partir de

bases de dados. Algumas dessas *shells*, como o MSBNx[4] da *Microsoft*, são responsáveis somente por realizar inferências; outras, como o *Hugin Developer*[5], não são *freewares.*

Dentre as pesquisadas, que foram relacionadas no item 5.3, destacam-se três *shells*, por possuírem API para aquisição de conhecimento, que serão descritas a seguir: o *Hugin Lite*[6], o BNPC[7] e o UnBBayes[8].

7.1.1 Hugin Lite

O *Hugin Lite* é a versão gratuita, disponível para várias plataformas, oferecida pela empresa *Hugin Expert*, que inclui um ambiente gráfico que auxilia o usuário na criação das redes, e o *Hugin Decision Engine* (HDE), que contém quatro tipos de interfaces para ambientes de programação, ou seja, API para C, C++, Java e *Visual Basic*, e também seus manuais de uso. Em ambos, é possível construir uma RB manualmente ou por meio da aprendizagem a partir de bases de dados.

As API do *Hugin* permitem que programadores possam construir aplicações baseadas em conhecimento, por meio do uso de suas bibliotecas, oferecendo, assim, recursos de aprendizagem e inferência no desenvolvimento de SEP.

Nessa *shell,* para a construção de uma RB via API, utiliza-se o algoritmo PC[9] para aprendizagem da estrutura da rede e o algoritmo EM[9] para definição da distribuição das probabilidades condicionais.

Na linguagem C, a API dessa *shell* chama-se *hugin.dll*, que deve ser declarada juntamente com a biblioteca *hugin.h* para que seja possível utilizar essa DLL.

---

[4] Disponível em: http://research.microsoft.com/adapt/MSBNx.
[5] Pacote específico para desenvolvedores, disponível em: http://www.hugin.com.
[6] Disponível em: http://www.hugin.com.
[7] Disponível em: http://www.cs.ualberta.ca/~jcheng/bnpc.htm.
[8] Disponível em: http://unbbayes.sourceforge.net.
[9] Algoritmo baseado no método de busca e pontuação.

Basicamente sua utilização consiste nas seguintes funções, que devem ser chamadas respectivamente:

a) *h_domain_t h_load_domain(h_string_t)*: responsável por abrir uma base de dados, de onde será descoberto o conhecimento,

b) *h_status_t h_domain_learn_structure(h_domain_t)*: função que realiza a aprendizagem da estrutura;

c) *h_status_t h_domain_learn_tables(h_domain_t)*: realiza a aprendizagem das probabilidades condicionais;

d) *h_status_t h_domain_save_as_net(h_domain_t,h_string_t)*: exporta a rede criada.

Embora o *Hugin* disponha de uma série de manuais sobre o uso de suas API, não fica compreensível, na versão *Lite*, a maneira como elas devem ser chamadas, de que forma os parâmetros devem ser passados para as funções e como tratar a base de dados.

Durante o estudo dessa API, entrou-se em contato com o fabricante da *shell*, a *Hugin Expert*, para ver se a mesma dispunha de um material de referência mais detalhado sobre a utilização das API para a aquisição de conhecimento em bases de dados. A empresa retornou, relatando que toda a documentação gratuita encontrava-se no HDE. Entrou-se em contato também com a única representante brasileira da empresa, mas a mesma não se manifestou favorável à pesquisa, justificando que seu objetivo é comercial e não acadêmico.

A segunda *shell* de mineração de dados em RB estudada foi o BNPC.

7.1.2 BNPC

O BNPC é utilizado na plataforma *Windows*, composto de um assistente gráfico que auxilia o usuário na escolha das configurações referentes à base de dados no processo de construção da RB, e de uma API. Foi desenvolvido no Canadá, por Jie Cheng, pesquisador do Departamento de Ciência da Computação da Universidade de Alberta.

A API do BNPC é uma *ActiveX* [10] DLL, que permite sua integração a outras ferramentas que trabalham com RB, mineração de dados, sistemas baseados em conhecimento e ambientes de desenvolvimento (CHENG, 2006).

Via API, para a construção de uma RB, utiliza-se o algoritmo CBL[11], disponível nas versões A e B, para a aprendizagem da estrutura e das probabilidades.

Basicamente, a utilização da API inicia-se referenciando-a em um novo projeto a ser desenvolvido. A seguir, cria-se um objeto *dataset* e são chamadas as funções *Init()*, *Construct()* e *LearnCP()*, responsáveis, respectivamente, pela inicialização, construção e exportação da RB.

Entrou-se em contato com o fabricante da *shell*, para ver se o mesmo disponibiliza material extra sobre sua API, e obteve-se, como retorno, que toda documentação existente é encontrada em seu *site*.

A terceira *shell* analisada, por possuir API para aquisição de conhecimento em bases de dados, foi o UnBBayes.

[10] Conjunto de tecnologias que facilitam a integração a diversos tipos de aplicação.
[11] Algoritmo baseado no método de análise de dependência.

7.1.3 UnBBayes

A *shell* UnBBayes é um *software* aberto, desenvolvido em Java pelo Grupo de Pesquisa em Inteligência Artificial do Departamento de Ciência da Computação da Universidade de Brasília, disponibilizado gratuitamente para fins não comerciais, independente de plataforma, para a construção de RB. É composta de um ambiente visual, uma máquina de inferências e de uma API.

A API do UnBBayes é uma biblioteca de classes *.class* escrita também em Java, e sua documentação é encontrada no *help* da *shell*, onde é possível analisar suas funcionalidades.

Via API, para a construção de uma RB, são utilizados os algoritmos K2[12], B[12] e CBL[13] nas versões A e B, sendo que cada um deles, é responsável pela aprendizagem da estrutura e das probabilidades. Esses algoritmos estão implementados dentro do *package*[14] unbbayes.aprendizagem fornecido pela API. Nesse *package*, existem quatro classes que implementam funções de acordo com os algoritmos citados. As funções referentes a cada algoritmo são:

a) K2(NodeList variables, byte[][] dataBase, int[] vector, long caseNumber, java.lang.String metric, java.lang.String param, boolean compacted);
b) B(NodeList variables, byte[][] dataBase, int[] vector, long caseNumber, java.lang.String metric, java.lang.String param, boolean compacted);
c) CBLA(NodeList variables, byte[][] dataBase, int[] vector, long caseNumber, java.lang.String param, boolean compacted);
d) CBLB(NodeList variables, byte[][] dataBase, int[] vector, long caseNumber, java.lang.String param, boolean compacted).

[12] Algoritmo baseado no método de busca e pontuação.
[13] Algoritmo baseado no método de análise de dependência.
[14] Em Java, é uma maneira de agrupar classes de objetivos afins ou de uma mesma aplicação.

Essas funções recebem como parâmetros uma lista de variáveis, e uma matriz com os dados do arquivo da base de dados, e a partir disso, monta a RB correspondente.

Entrou-se em contato com o responsável na Universidade de Brasília pela parte de aprendizagem da *shell*, para ver se o mesmo disponibilizava uma documentação mais precisa sobre como integrar sua API a um ambiente de desenvolvimento, e obteve-se como retorno, que todo material existente, encontra-se no *help* da *shell*.

A seguir na Figura 7, é realizada uma comparação entre as API das *shells* analisadas, que fazem aprendizagem, onde são mostradas as suas características referentes a disponibilidades de linguagens, realização de inferências, complexidade de uso, documentação, realização de aprendizagem a partir de bases de dados, quantidade de algoritmos implementados, entendimento dos parâmetros das funções e quantidade de funções.

CARACTERISTICAS DAS API ANALISADAS	***HUGIN LITE***	**BNPC**	**UNBBAYES**
Shell freeware	Sim	Sim	Sim
Quantidade de API	Quatro	Uma	Uma
Linguagens disponibilizadas pela API	C, C++, Java, *Visual Basic*	Linguagens 32 *bits*	Java
Presença de funções na API para inferências	Sim	Não	Sim
Complexidade no entendimento do uso da API	Mais complexa	Menos complexa	Mais complexa
Realização de aprendizagem a partir de bases de dados	Sim	Sim	Sim
Documentação da API	Manuais	*Site*	*Help*
Documentação restrita sobre a utilização das funções da API para aprendizagem a partir de bases de dados	Sim	Sim	Sim
Quantidade de funções presentes na API	Muitas	Três	Muitas
Especificação de todos os parâmetros passados às funções da API	Pouca	Sim	Pouca
Quantidade de algoritmos para aprendizagem a partir de bases de dados	Dois	Um	Três
Algoritmos disponibilizados para aprendizagem a partir de bases de dados	PC, EM	CBL (versões A e B)	B, K2, CBL (versões A e B)

Figura 7. Comparação entre as API das *shells*

De acordo com o estudo dessas *shells*, verificou-se que a documentação referente à utilização de suas API para a aquisição de conhecimento a partir de bases de dados foi muito restrita em todas as *shells.*

Sendo assim, pelas características apresentadas, foi selecionada uma *shell* para a integração, conforme será visto a seguir.

7.2 SELEÇÃO DE UMA *SHELL* E DE UM AMBIENTE DE DESENVOLVIMENTO

Dentre as API estudadas, foi escolhida a API do BNPC para a integração a um ambiente de desenvolvimento, primeiro, porque Manarin (2004) a descreve como a *shell* que melhor gerou a RB para a base de dados escolhida no seu domínio de aplicação; segundo, pelo fato das outras *shells* não disponibilizarem material de referência específico sobre o uso de suas API; terceiro, por sua documentação mostrar, de maneira mais detalhada, cada parâmetro que deve ser passado para suas funções; e quarto, porque Koehler et al (2004) descreve que o algoritmo do BNPC, descrito no Apêndice A, é um dos mais eficientes e ágeis para a aprendizagem de RB a partir de bases de dados.

Ressalta-se que o BNPC também é escasso de referência, sendo que o único material existente sobre sua API encontra-se no *site* da *shell*, onde sua utilização é descrita de maneira simplificada.

Embora o fabricante tenha testado a API do BNPC usando apenas *Visual Basic*, o mesmo considera possível sua integração com qualquer outra linguagem de programação 32 *bits* (CHENG, 2006).

Nos testes realizados com os ambientes de desenvolvimento C++ Builder e Delphi, não foi possível realizar sua integração com a API do BNPC, pelo fato da conexão com a base de dados ser realizada utilizando um mecanismo denominado *Data Access Objects* (DAO), e de ser muito restrita a documentação dessas linguagens utilizando esse tipo de conexão.

Dessa forma, optou-se pela linguagem de programação *Visual Basic* para a construção do protótipo, pois, além de ser recomendada pelo fabricante, aceita esse tipo de conexão e dispõe de exemplos de como fazer esse acesso a dados.

Considerando a utilização do BNPC, a seguir serão detalhados os recursos de sua API para a aquisição automatizada de conhecimento, e conseqüente construção de RB a partir de bases de dados.

7.3 ESTUDO DOS RECURSOS DA API DO BNPC

Os recursos disponíveis pela API do BNPC, denominada BNPCAPI.DLL, estão concentrados em três funções – *Init(), Construct() e LearnCP()* –, responsáveis pela aprendizagem estrutural e probabilística de uma RB.

A primeira delas, a função *Init()*, inicializa a construção da rede, e é composta de alguns parâmetros, descritos abaixo:

a) **dbobj:** representa a conexão ao banco de dados por meio de um objeto DAO, que pode ser um DAO/JET[15] ou DAO/ODBCDirect[16];

b) **txttbnm:** é o nome da tabela do banco de dados;

c) **intnumflds:** representa o número de campos da tabela que serão utilizados na construção da RB;

[15] Permite acesso a fontes de dados externas.
[16] Permite acesso aos bancos de dados por meio de ODBC.

d) **fldnames():** é um vetor do tamanho do parâmetro anterior (intnumflds) e contém o nome dos campos escolhidos;

e) **txtfrqfld:** se os dados da tabela estiverem no formato compactado[17], esse parâmetro recebe o nome do campo que contém as freqüências; caso contrário, esse parâmetro não recebe nome nenhum;

f) **bco:** esse parâmetro recebe verdadeiro, caso os campos escolhidos para a geração da RB, localizados no vetor fldnames() estiverem na ordem correta de dependência, ou falso, se não estiverem ordenados;

g) **arrpo():** representa uma matriz bidimensional quadrada na forma arrpo(a,b) do tamanho do parâmetro intnumflds especificado anteriormente. Quando “a” for causa indireta de “b”, ou “a” aparecer antecipadamente em relação a “b” na ordenação de fldnames(), essa posição da matriz recebe valor 1; caso contrário, recebe valor 0;

h) **arrce():** também representa uma matriz bidimensional quadrada na forma arrce(a,b) do tamanho do parâmetro intnumflds. Quando “a” for causa direta de “b”, essa posição da matriz recebe valor 1; quando não for, recebe valor 0;

i) **arrfl():** representa uma matriz bidimensional quadrada na forma arrfl(a,b) de tamanho intnumflds. Esse parâmetro refere-se aos *links* proibidos, isto é, relações que não devem acontecer entre os campos escolhidos. Quanto “a” não deve se relacionar com “b”, essa posição na matriz recebe 1; caso contrário, recebe valor 0;

j) **root():** é um vetor do tamanho do parâmetro intnumflds e deve ser preenchido conforme a ordem do vetor que contém os nomes dos campos

[17] Formato onde cada registro possui uma variável que informa a sua freqüência, com o intuito de evitar ocorrências múltiplas.

escolhidos da tabela (fldnames()). Assim, os campos que devem ser nós raízes da RB, devem receber valor 1 no vetor root() na posição correspondente ao vetor fldnames(); caso contrário, recebem valor 0;

k) **leaf():** também é um vetor de tamanho intnumflds e funciona da mesma forma que o parâmetro anterior, mas dessa vez, defini-se os nós folhas da RB por meio do valor 1 no campo correspondente; caso o campo não for folha, recebe valor 0;

l) **txtinputlog:** é um parâmetro opcional e refere-se a um arquivo de *log* criado anteriormente pela *shell* ou pela API (no parâmetro seguinte), onde são armazenadas informações a respeito das configurações que levaram a construção de uma RB, visando acelerar o processo de criação de uma nova;

m) **txtoutputlog:** é também um parâmetro opcional que salva e armazena num arquivo de *log*, informações a respeito do processo de criação da RB, que poderá ser usado no parâmetro anterior.

A segunda função da API, definida como *Construct()*, é responsável por construir a RB, de acordo com os dados passados para a função *Init()*. Ela é composta de três parâmetros:

a) **times_of_default:** representa o valor do *threshold*[18]. A *shell* traz como *default* o valor 1.0, mas os valores vão de 0.1 a 10.0. Geralmente, segundo o fabricante, não é preciso mudar essa configuração, pois o valor *default* garante bons resultados. Na API esse valor também pode ser configurado;

[18] Valor de corte, que representa o tipo de relação (forte ou fraca) a ser encontrada entre as variáveis

b) **bpreserv:** esse parâmetro recebe verdadeiro, caso as relações de causa e efeito definidas pelo usuário devam ser preservadas, ou falso, caso possam ser descartadas;

c) **graph():** representa uma matriz bidimensional quadrada na forma graph (a,b) do tamanho do parâmetro intnumflds especificado na função anterior. Esse parâmetro é quem recebe a RB criada, e é nele que devem ser realizadas mudanças caso haja a necessidade de criar, trocar ou anular uma relação de um nó a outro da rede. Para se alterar a rede, têm-se as seguintes situações: se “a” não tiver relação direta com “b”, essa posição da matriz recebe valor 0; se “a” for pai de “b”, então essa posição da matriz recebe valor 1; se “b” for pai de “a”, essa posição da matriz recebe valor 2; e se “a” e “b” tem relação direta, mas a direção não pode ser decidida, a posição da matriz onde esse caso acontece recebe valor 3.

A terceira e última função da API chama-se *LearnCP()* e é responsável por exportar a RB para um arquivo, com base no que foi definido nas funções anteriores. Ela é composta de dois parâmetros:

a) **graph():** o mesmo parâmetro da função anterior, que representa a RB criada e possivelmente alterada, caso tenha sido necessário;

b) **txtfilenm:** representa o nome do arquivo para o qual a RB será salva. A API suporta os formatos (.net) para o *Hugin Expert*, (.dne) para o Netica e (.bif) para o *Bayesian Interchange Format*.

Após a chamada dessas três funções, a API gera automaticamente a RB, com suas partes quantitativa e qualitativa, a partir da passagem dos parâmetros, e a salva em um arquivo de acordo com a extensão escolhida. A seguir, a rede pode ser visualizada

em uma *shell* para inferências em RB, como o Netica[19] ou o *Hugin*, de acordo com o formato salvo.

A seguir, será realizada uma comparação entre os recursos da API e as etapas que a *shell* disponibiliza em seu assistente gráfico.

7.3.1 Comparação entre a *shell* e a API

Para o entendimento do funcionamento dos recursos da API, foi necessário estudar o processo de construção da rede pela *shell*, pelo fato dos nomes e da seqüência dos parâmetros passados para as três funções da API, se relacionarem com as etapas que a *shell* disponibiliza em seu assistente gráfico.

Na comparação, adotou-se a ordem *shell*➔API, ou seja, mostra-se como determinada ação é realizada no BNPC e, em seguida, como essa mesma ação acontece ou é tratada pela API, para que as suas funções sejam melhor compreendidas.

O assistente gráfico do BNPC é composto de cinco etapas principais que ajudam o usuário na construção da RB. Os três primeiros passos referem-se a conexão à base de dados e a escolha dos campos que deverão ser usados na etapa seguinte, que a *shell* denomina de domínio do conhecimento, onde serão feitos os ajustes e relações com essas variáveis selecionadas. No passo final, pode-se salvar um arquivo de *log* com as informações das configurações usadas durante o processo ou importar um já salvo.

A Figura 8, ilustra a tela de apresentação da *shell*, onde o usuário pode ter informações a respeito da versão, do funcionamento, entre outras.

[19] Disponível em: http:// www.norsys.com.

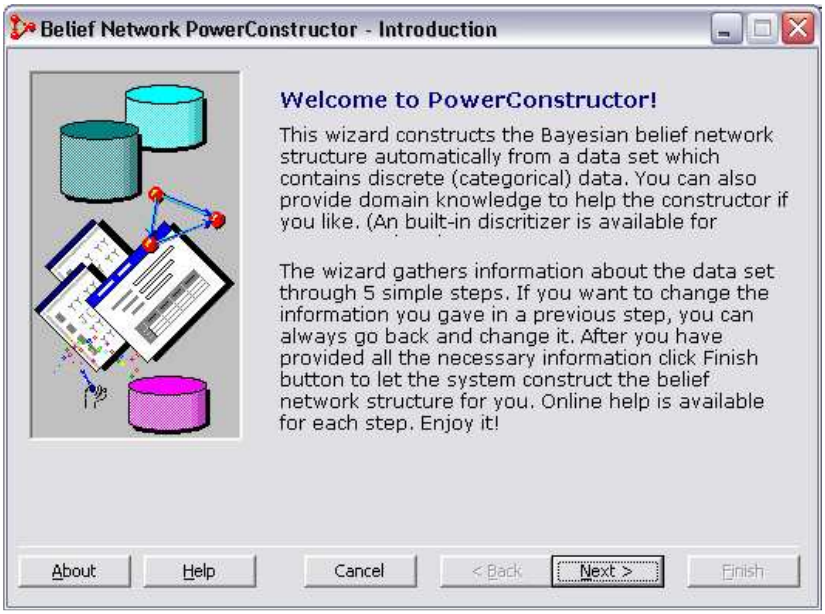

Figura 8. Tela de apresentação do BNPC
Fonte: CHENG, J. (2006)

O primeiro passo da *shell* (Figura 9) representa a escolha do formato em que a base de dados se encontra.

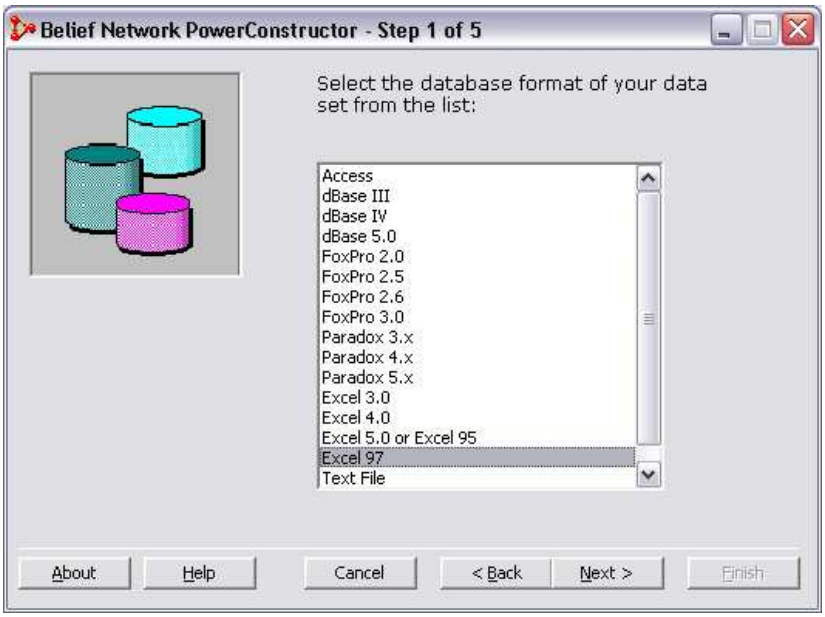

Figura 9. Tela de apresentação do BNPC
Fonte: CHENG, J. (2006)

A seguir, no segundo passo, especifica-se o caminho onde o arquivo se encontra, conforme mostra a Figura 10.

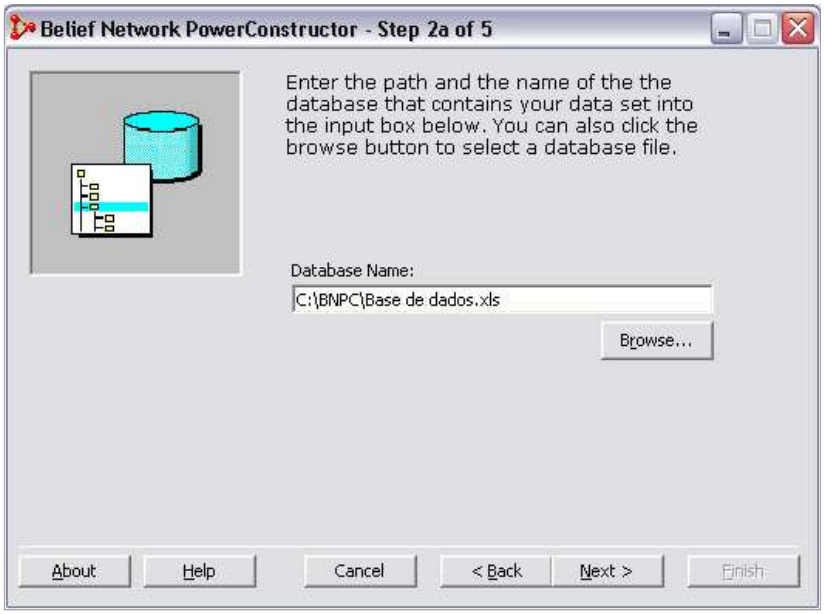

Figura 10. Caminho da base de dados
Fonte: CHENG, J. (2006)

Via API do BNPC, esses dois passos são realizados utilizando um mecanismo de conexão a banco de dados DAO, DLL que vem junto com o *Microsoft Office*, e que permite o uso de uma linguagem de programação para acessar e manipular dados em bancos de dados locais ou remotos. O DAO fornece suporte a dois tipos diferentes de ambientes (ou espaços de trabalho) de bancos de dados: o *Microsoft Jet*[20], que permite acesso a fontes de dados externas, como por exemplo, *Excel*, *Access*, *Paradox*, por meio de um *driver* chamado *Indexed Sequential Access Method*[21] (ISAM); e o *ODBCDirect*[22] que permite acesso aos servidores de bancos de dados por meio de ODBC (OFFICE, 2006).

Dessa forma, existem duas formas de conexão DAO, e diversos objetos que devem ser tratados para trabalhar no tipo de acesso escolhido. Na API, esse objeto DAO é o primeiro parâmetro passado para a função *Init()*, conforme mostra a Figura 11.

[20] Um sistema de gerenciamento de bancos de dados que recupera e/ou armazena dados de um banco de dados do usuário ou do sistema.
[21] Um *driver* que pode ser especificado, localizado no registro do *Windows* e que contém os formatos dos bancos de dados externos.
[22] Acesso direto a ODBC, sem utilizar o *Microsoft Jet*

```
Init(dbobj, txttbnm, intnumflds, fldnames, txtfrqfld, bco, arrpo,
     arrce, arrfl, root, leaf, txtinputlog, txtoutputlog)
```

Figura 11. Parâmetro que representa o objeto da conexão DAO
Fonte: CHENG, J. (2006)

Ressalta-se que o objetivo desse trabalho é explorar os recursos de API da *shell*, e não concentrar-se na questão do acesso aos bancos de dados possíveis de serem conectados utilizando esse mecanismo DAO.

No terceiro passo (Figura 12), escolhe-se o conjunto de dados (tabela) da base de dados, e os campos da mesma que serão usados na próxima etapa. Na API, conforme Figura 13, essas configurações são representadas pelos parâmetros txttbnm, intnumflds, fldnames(), txtfrqfld, que referem-se, respectivamente, ao nome da tabela, por meio da conexão a base de dados pelo objeto DAO, realizada pelo parâmetro dbobj; ao número inteiro adquirido pela contagem dos campos selecionados da tabela; ao vetor contendo o nome dos campos escolhidos; e ao nome do campo que contém as freqüências, caso a tabela da base de dados esteja no formato compactado.

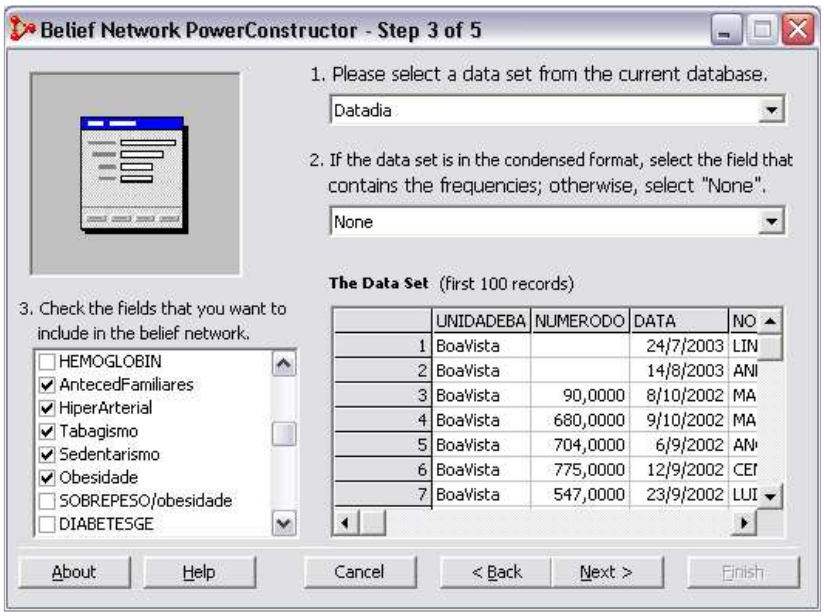

Figura 12. Escolha da tabela e dos campos
Fonte: CHENG, J. (2006)

```
Init(dbobj, txttbnm, intnumflds, fldnames, txtfrqfld, bco, arrpo,
     arrce, arrfl, root, leaf, txtinputlog, txtoutputlog)
```

Figura 13. Parâmetros nome da tabela, número de campos, vetor de nomes, variável das freqüências
Fonte: CHENG, J. (2006)

O quarto passo para a construção da RB usando o BNPC consiste no que a *shell* denomina de domínio do conhecimento, onde os campos selecionados na etapa anterior são relacionados entre si. Pela *shell* é possível avançar essa etapa não especificando nada nela e, nesse caso, a RB gerada ficará sem *links* entre os campos. No editor gráfico, na parte final do assistente, onde é demonstrada a parte qualitativa da rede, é possível atribuir relações entre os nós existentes. Pela API, é interessante passar os parâmetros corretos nessa etapa, para que não haja a necessidade de alteração nos links na função *Construct()*. Como esse passo é o mais importante na construção da RB, ele deve ser bem analisado.

A Figura 14 ilustra na *shell*, onde devem ser especificados os nós raízes e folhas e, via API, conforme ilustra a Figura 15, a identificação desses nós é realizada com o preenchimento correto dos vetores root() e leaf(), respectivamente, raízes e folhas.

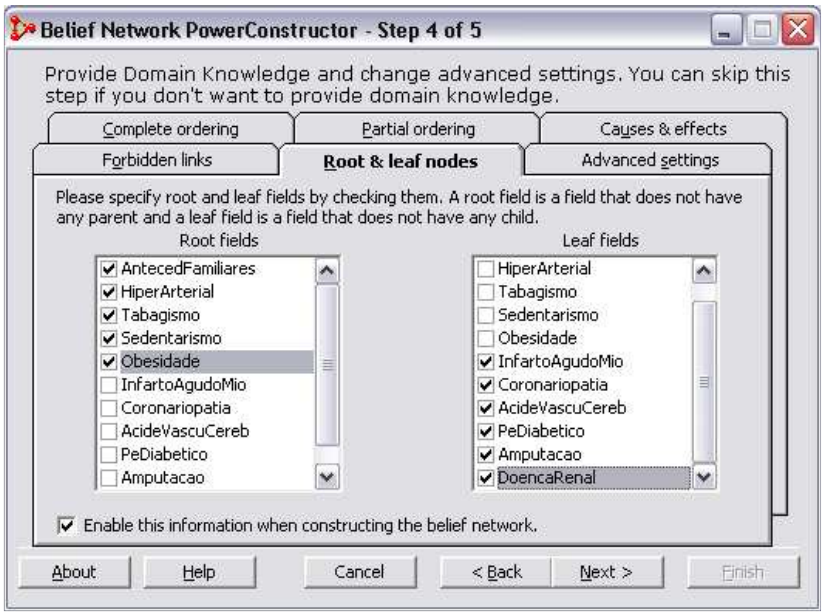

Figura 14. Escolha dos nós raízes e folhas
Fonte: CHENG, J. (2006)

```
Init(dbobj, txttbnm, intnumflds, fldnames, txtfrqfld, bco, arrpo,
     arrce, arrfl, root, leaf, txtinputlog, txtoutputlog)
```

Figura 15. Parâmetros que representam as matrizes raízes e folhas
Fonte: CHENG, J. (2006)

A Figura 16 especifica a relação de causa e efeito que o usuário pode definir entre os nós definidos como raízes e folhas. A Figura 17 mostra que, pela API, essa especificação é realizada com o preenchimento da matriz arrce().

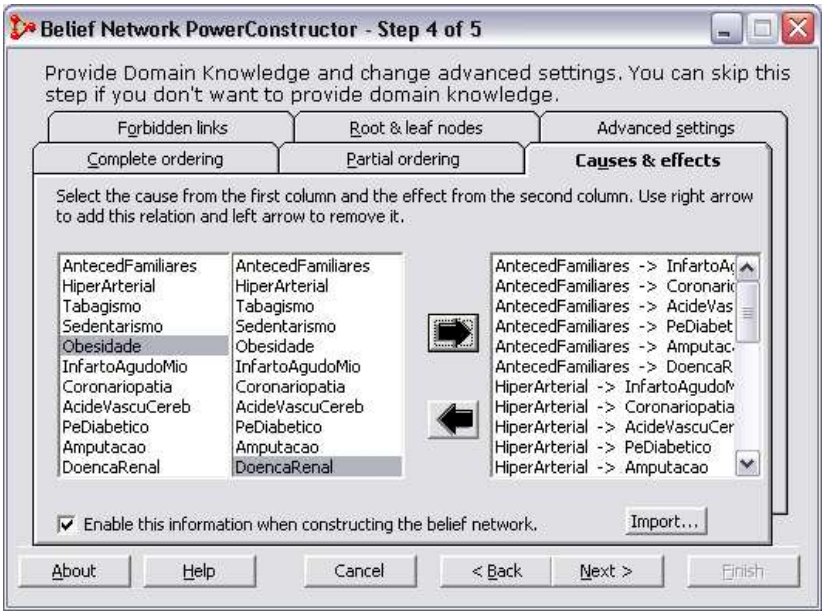

Figura 16. Relação de causas e efeitos
Fonte: CHENG, J. (2006)

```
Init(dbobj, txttbnm, intnumflds, fldnames, txtfrqfld, bco, arrpo,
     arrce, arrfl, root, leaf, txtinputlog, txtoutputlog)
```

Figura 17. Parâmetro que representa a matriz de causas e efeitos
Fonte: CHENG, J. (2006)

Na *shell*, as configurações avançadas, ilustradas pela Figura 18, são representadas na API (Figura 19) pelos dois parâmetros iniciais que compõe a segunda função (*Construct()*) a ser chamada, sendo que, o primeiro, times_of_default, especifica o valor do *threshold*, que por padrão é sempre 1.0, e o segundo, bpreserv, preserva ou não as relações de causa e efeito definidas pelo usuário. Salienta-se que, se a matriz arrce() estiver preenchida, ou seja, o usuário definiu causas e efeitos, é necessário que o

parâmetro bpreserv receba verdadeiro, para que essas relações sejam preservadas na construção da rede; caso contrário, a RB, via API, não preservará as causas e efeitos definidos pelo usuário, podendo ficar com mais, menos ou sem todos os *links* que foram definidas no arrce().

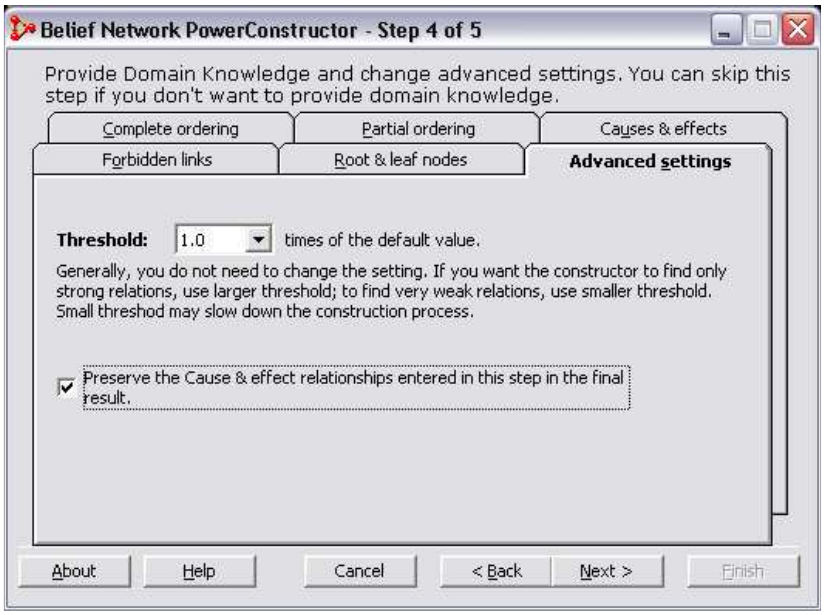

Figura 18. Configurações avançadas
Fonte: CHENG, J. (2006)

```
Construct(times_of_default, bpreserv, graph)
```

Figura 19. Parâmetros *threshold* e preservação de causas e efeitos
Fonte: CHENG, J. (2006)

As demais opções desse passo do assistente gráfico da *shell*, onde se define o domínio do conhecimento, não precisam necessariamente ser preenchidas para a construção de uma RB. Via API, essas opções referem-se ainda aos parâmetros da função *Init()*, ilustrada pela Figura 20. A parte de ordenação completa é tratada pelo parâmetro bco; a ordenação parcial, é representada pela matriz arrpo(); e a última opção desse quarto passo, refere-se a questão dos *links* proibidos pelo preenchimento da matriz arrfl().

```
Init(dbobj, txttbnm, intnumflds, fldnames, txtfrqfld, bco, arrpo,
     arrce, arrfl, root, leaf, txtinputlog, txtoutputlog)
```

Figura 20. Parâmetros referentes à ordenação completa, ordenação parcial e *links* proibidos
Fonte: CHENG, J. (2006)

Caso a matriz arrfl() venha a ser usada, representando aqueles nós onde não devem haver *links* entre si, deve-se tomar cuidado para que essa relação a➔b não esteja conflitando com a matriz arrce(), que representa as causas e efeitos. Assim, o que for 1 em uma relação a➔b em uma matriz, não pode ser na outra, e vice-versa, ou seja, se duas variáveis tiverem *links* proibidos, elas não devem ter relação de causa e efeito, pois caso essa relação seja confirmada, essa situação ocasionará um erro de ligação entre elas na construção da rede.

O quinto e último passo do assistente do BNPC (Figura 21) refere-se à criação ou importação de um arquivo de log já salvo anteriormente, contendo informações das configurações realizadas até aqui na *shell*. Na API (Figura 22), essas opções representam os dois últimos parâmetros da chamada *Init()*, txtinputlog e txtoutputlog, responsáveis pela importação e criação de um *log*, respectivamente.

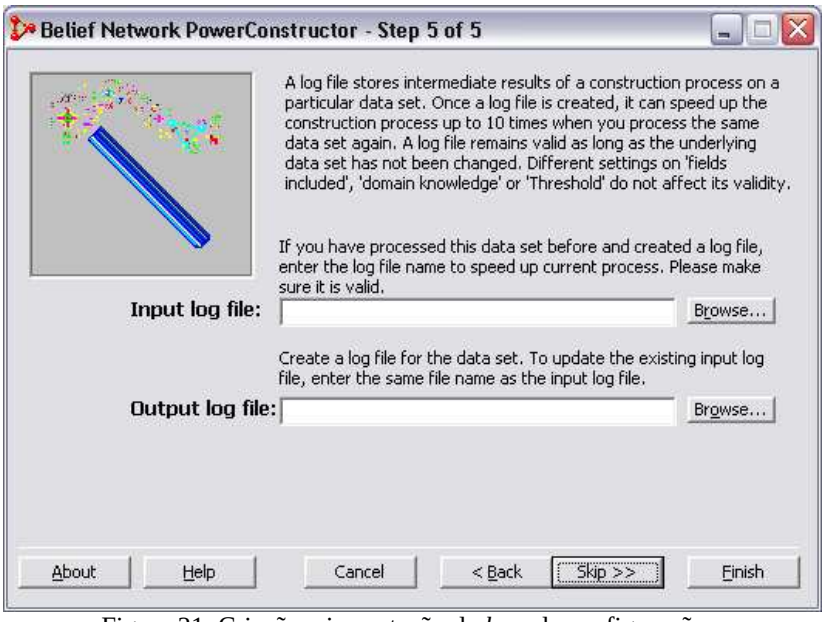

Figura 21. Criação e importação de *logs* de configuração
Fonte: CHENG, J. (2006)

```
Init(dbobj, txttbnm, intnumflds, fldnames, txtfrgfld, bco, arrpo,
     arrce, arrfl, root, leaf, txtinputlog, txtoutputlog)
```

Figura 22. Parâmetros que representam criação e importação de logs
Fonte: CHENG, J. (2006)

Após a utilização do assistente gráfico da *shell* BNPC, apresenta-se uma tela (Figura 23), mostrando que o processo foi finalizado. Existem algumas opções presentes nessa parte, que se referem a exportar a rede criada para um dos formatos descritos anteriormente, e ao editor gráfico, que pode ser aberto para a visualização dos nós e das relações entre eles, bem como a realização de possíveis alterações nos *links* da RB criada, antes da mesma ser exportada. Na API (Figura 24), qualquer alteração nos *links* – após a chamada da função *Init()* – pode ser realizada na matriz graph(), que é onde a RB ficará até ser chamada a função *LearnCP()*, para salvar a rede.

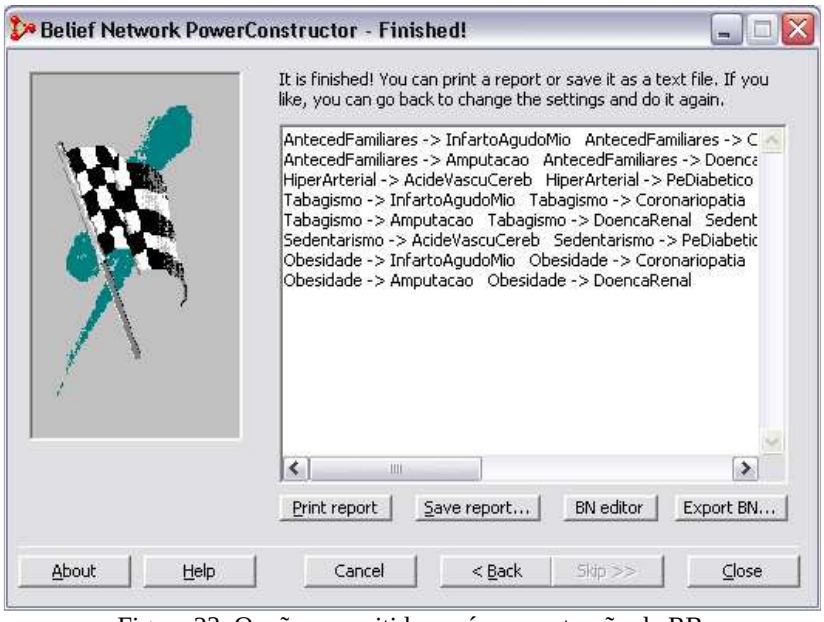

Figura 23. Opções permitidas após a construção da RB
Fonte: CHENG, J. (2006)

```
Construct(times_of_default, bpreserv, graph)
```

Figura 24. Parâmetro responsável pela alteração de *links* na rede
Fonte: CHENG, J. (2006)

A Figura 25 mostra a tela do BNPC responsável pela exportação da RB gerada, sendo possível definir um nome, o formato, e o local onde o arquivo será salvo. Via API, conforme ilustra a Figura 26, a exportação da rede é realizada chamando a função *LearnCP()*, com os parâmetros graph() e txtfilenm, sendo esse último, o nome do arquivo e sua extensão.

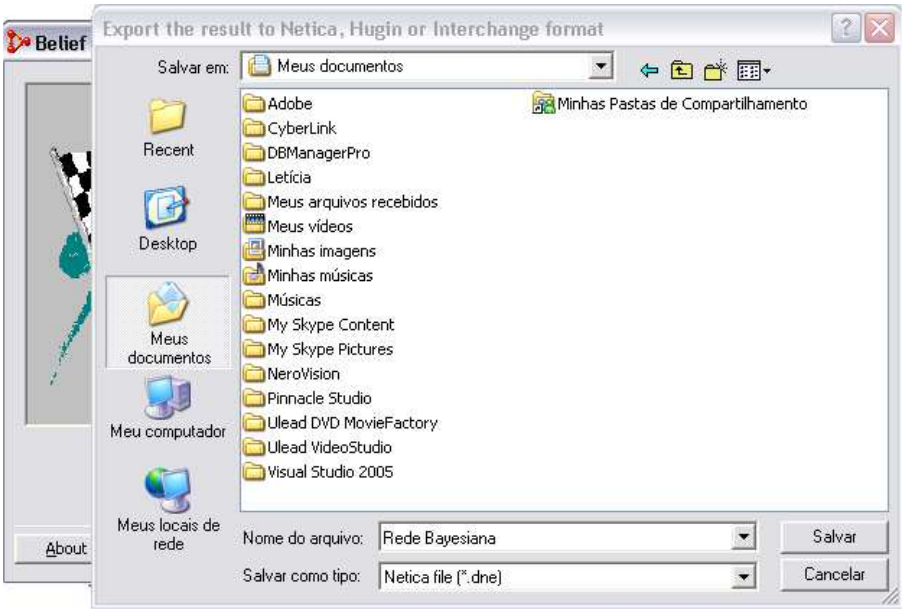

Figura 25. Exportação da RB
Fonte: CHENG, J. (2006)

```
LearnCP(graph, txtfilenm)
```

Figura 26. Parâmetros responsáveis por exportar a RB
Fonte: CHENG, J. (2006)

Dessa forma, definida a API e realizado o estudo dos recursos disponibilizados para aprendizagem, a seguir, defini-se uma base de dados e realiza-se a integração ao ambiente de desenvolvimento.

7.4 DEFINIÇÃO DE UMA BASE DE DADOS E INTEGRAÇÃO DO AMBIENTE DE DESENVOLVIMENTO COM A API DO BNPC

A base de dados definida para a avaliação do protótipo foi a mesma utilizada por Manarin (2004) para a aquisição de conhecimento, e relaciona os fatores de risco à determinadas complicações em pacientes com Diabetes Mellitus tipo dois nos bairros de Rio Maina, São Sebastião, Mineira Velha, Renascer, Laranjinha, São Luiz e Boa Vista, da cidade de Criciúma.

De acordo com as etapas do KDD, foi definida essa base de dados pelo fato dela já estar pré-processada, ou seja, seus dados já foram tratados, reduzidos e limpos, estando assim, pronta para ser minerada, e também por se tratar de uma base médica, propícia para análises por meio de RB, por representar um domínio de conhecimento com certo grau de incerteza.

O Diabetes Mellitus é uma das principais doenças crônicas que afetam o homem pela falta de insulina e/ou da incapacidade da insulina de exercer adequadamente seus efeitos, provocando o aumento de glicose no sangue (ORTIZ, ZANETTI, 2001). O tipo dois costuma ocorrer após os 35 anos de idade, embora possa manifestar-se em crianças e adolescentes com grave obesidade. Embora os pacientes possam ter complicações referentes ao excesso de açúcar no sangue, decorrentes do tratamento irregular do diabetes, os mesmos podem evitar essas complicações, pelo diagnóstico precoce e tratamento adequado (FILHO, 2001), (MILECH et al, 2002).

Os principais fatores de risco, segundo Czepielewski (2003), para o Diabetes Mellitus tipo dois são: uso de medicamentos que aumentam o nível de glicose, hereditariedade, hipertensão arterial, sedentarismo, doença coronariana (relacionada às artérias do coração) e Diabetes Mellitus gestacional prévio.

As principais complicações, conforme Milech et al (2002), são: infarto no coração, derrame cerebral, paralisia ou lesões nos olhos ou nervos, pé diabético, aumento da pressão arterial.

A base de dados é um arquivo *Microsoft Excel*, composto de 379 registros e 40 campos com informações dos pacientes cadastrados, referentes aos seus dados gerais, como nome, idade, sexo, nascimento; aos fatores de risco, como antecedentes familiares, hipertensal arterial, tabagismo, sedentarismo, classificação da obesidade; às complicações, como infarto agudo do miocárdio, coronariopatias, acidente vascular cerebral, pé diabético, amputação, doença renal; e aos medicamentos antidiabéticos utilizados. Essas informações foram cadastradas no arquivo, conforme preenchimento do protocolo de estudo no anexo A.

Após a definição da base de dados a ser utilizada para a realização de testes do protótipo, partiu-se para sua integração.

A integração foi realizada entre o ambiente de desenvolvimento *Microsoft Visual Basic 2005 Express Edition* e a versão 2.2 da API do BNPC, vinda com a *shell*.

O primeiro passo para a utilização da API é referenciá-la no projeto mostrando sua localização – no diretório onde o BNPC foi instalado – para que a aplicação a encontre e, assim, seja possível iniciar seu uso.

Em seguida, define-se um objeto *dataset*, representado como *MyDataSet*, para que, a partir dele, as três funções da API possam ser chamadas, conforme ilustra a Figura 27.

```
MyDataSet.Init(dbobj, txttbnm, intnumflds, fldnames, txtfrqfld, bco, arrpo,
               arrce, arrfl, root, leaf, txtinputlog, txtoutputlog)

           MyDataSet.Construct(times_of_default, bpreserv, graph)

                    MyDataSet.LearnCP(graph, txtfilenm)
```

Figura 27. Funções da API do BNPC

A próxima etapa é referenciar também no projeto a DAO360.DLL, pois essa API é responsável pela conexão DAO à base de dados por meio de uma série de funções pré-definidas, que podem ser analisadas e entendidas com a ajuda do seu *help*, que, assim como a DLL, vem com o *Microsoft Office*.

Como foi utilizada uma base de dados no formato *Excel*, a sua conexão foi realizada, via API do DAO, com o ambiente de bancos de dados *Microsoft Jet*, visto anteriormente.

Após esses procedimentos, foi necessário definir cada um dos parâmetros das três funções da API do BNPC, logo os vetores e matrizes foram preenchidos corretamente de acordo com os campos escolhidos da base de dados, e as variáveis foram valoradas. A seguir, chamou-se, conseqüentemente, as funções *MyDataSet.Init(),* *MyDataSet.Construct(),* e por fim, *MyDataSet.LeanCP(),* e a RB foi construída.

Para a compreensão do usuário, foi desenvolvida uma interface gráfica para o protótipo, denominado *VisionBayes*, que utiliza os recursos da API, para que o mesmo possa, a partir dos campos da base, escolher e definir os parâmetros, e assim, gerar a RB desejada.

A seguir, demonstra-se a utilização do protótipo *VisionBayes* para aquisição de conhecimento e os resultados obtidos.

7.5 RESULTADOS OBTIDOS

Com relação aos resultados obtidos pela integração da API da *shell* BNPC com o ambiente de desenvolvimento *Microsoft Visual Basic 2005 Express Edition* e a utilização da base de dados sobre o Diabetes Mellitus tipo dois em sete bairros de Criciúma, apresenta-se de acordo com a Figura 28, a interface gráfica do *VisionBayes* desenvolvida.

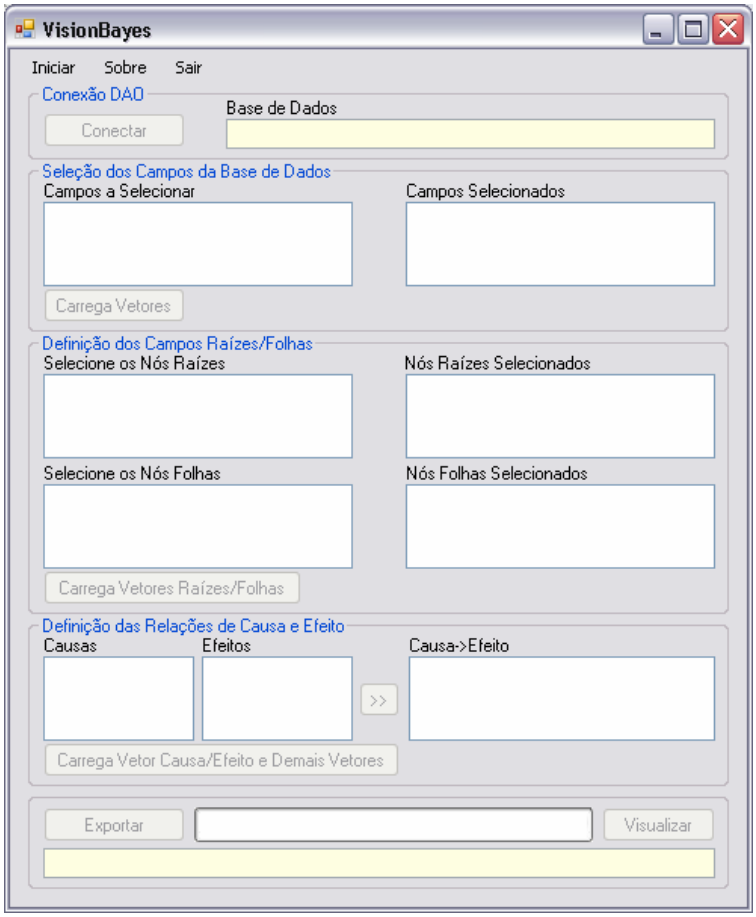

Figura 28. Tela do *VisionBayes*

Ao inicializar o *VisionBayes*, o usuário deve se conectar e escolher a base de dados, para que seus campos sejam mostrados na tela, e em seguida, ele possa selecioná-los.

A metodologia utilizada para a construção da RB pelo *VisionBayes* foi a mesma adotada por Manarin (2004), a fim de gerar a mesma rede, desta vez, via API do BNPC.

Considerando esse aspecto, na Figura 29, foram selecionados os campos referentes aos fatores de risco e complicações, ou seja, AntecedFamiliares, HiperArterial, Tabagismo, Sedentarismo, Obesidade, InfartoAgudoMio, Coronariopatia, AcideVascuCereb, PeDiabetico, Amputacao e DoencaRenal.

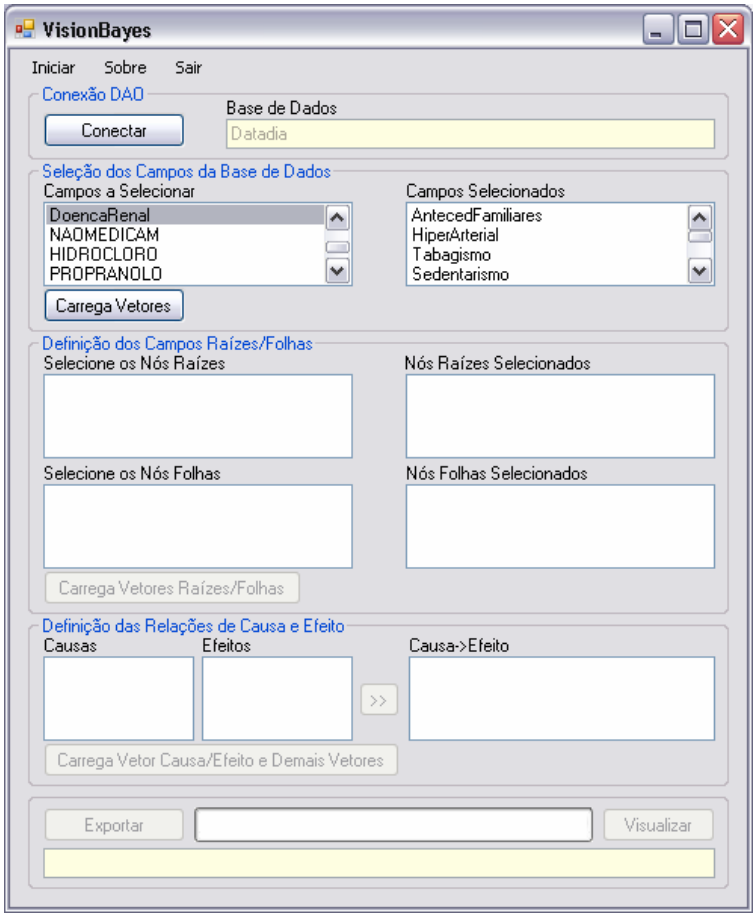

Figura 29. Seleção dos campos no *VisionBayes*

Em seguida, conforme Figura 30, carrega-se os vetores e são definidos os nós raízes, representados pelos fatores de risco, ou seja, AntecedFamiliares, HiperArterial, Tabagismo, Sedentarismo, Obesidade e os nós folhas, pelas complicações InfartoAgudoMio, Coronariopatia, AcideVascuCereb, PeDiabetico, Amputacao e DoencaRenal.

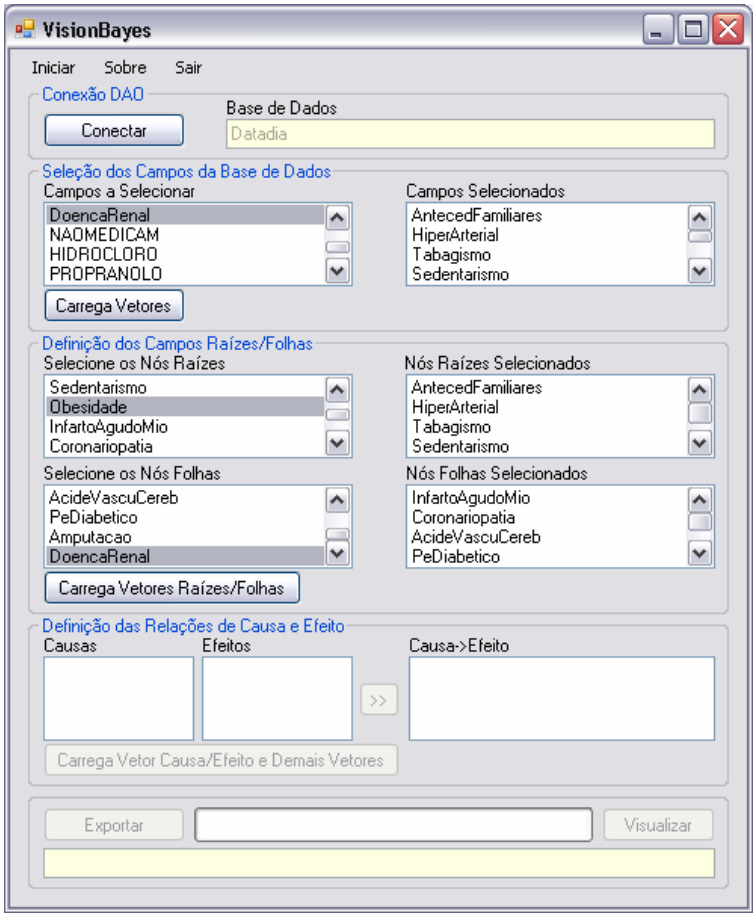

Figura 30. Definição das raízes e folhas no *VisionBayes*

A seguir são definidas as relações de causa e efeito (Figura 31) entre os nós raízes e os nós folhas, ou seja, todos os fatores de risco foram associados a todas as complicações.

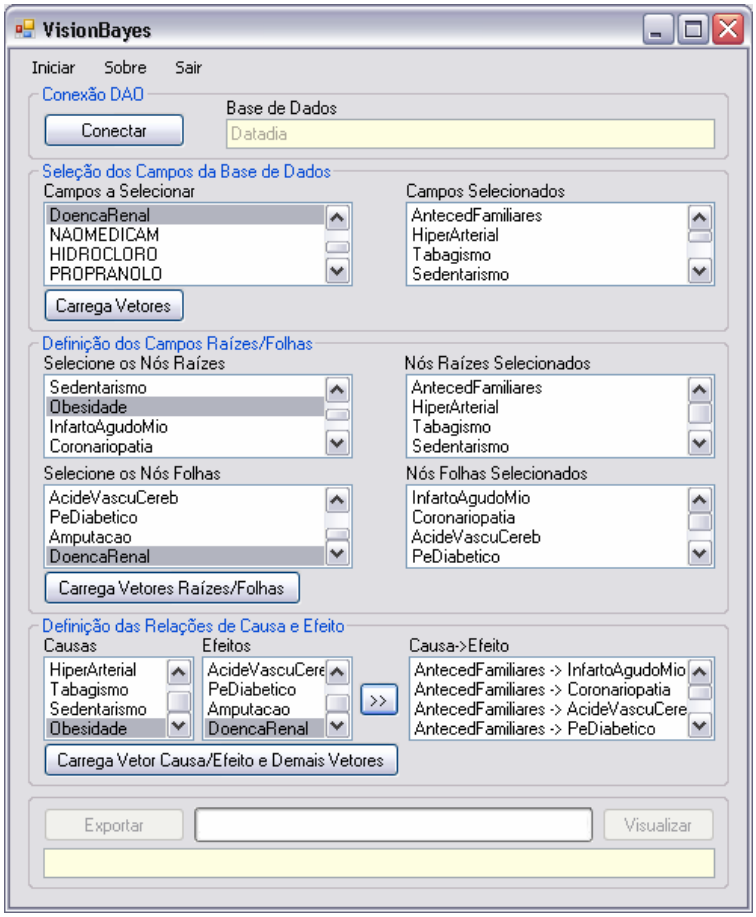

Figura 31. Definição das causas e efeitos no *VisionBayes*

A seguir realiza-se a aprendizagem, com base nas configurações definidas, e a RB é construída e exportada, podendo ser visualizada na *shell* Netica, como ilustra a Figura 32.

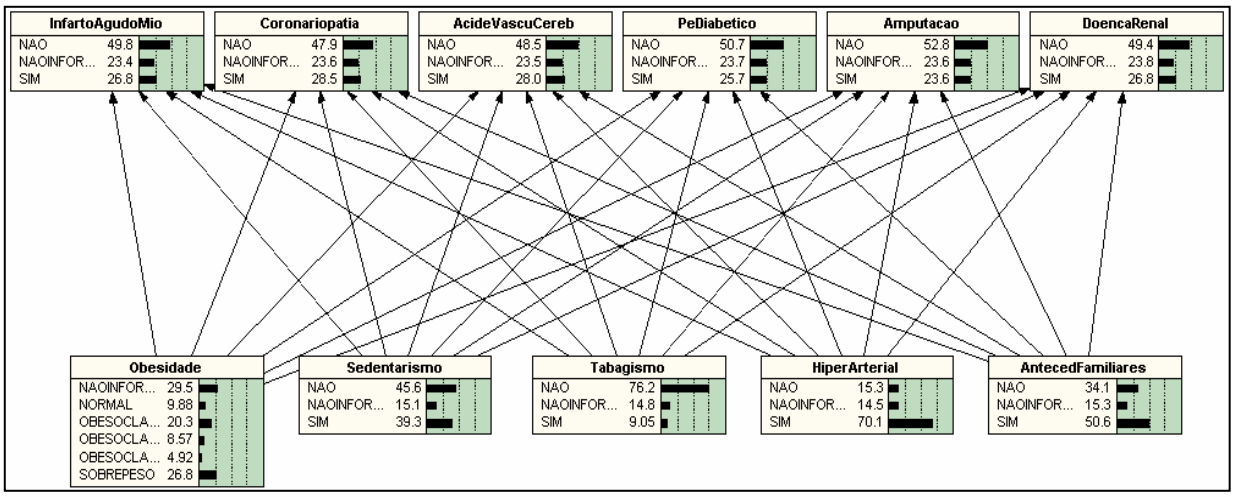

Figura 32. RB criada pelo *VisionBayes*

CONCLUSÃO

Esta pesquisa demonstrou a integração da API da *shell* de mineração de dados em RB denominada BNPC para a aquisição de conhecimento automatizada em SEP, a partir de bases de dados, resultando na construção do protótipo *VisionBayes*.

Na elaboração do trabalho, foi compreendido o processo que envolve o KDD, mais especificamente o aplicado as RB.

Foram analisados os recursos de API de três *shells freewares* de mineração de dados em RB, sendo elas, o *Hugin Lite*, o BNPC e o UnBBayes e, em todas, encontrou-se dificuldades relacionadas ao entendimento de suas funcionalidades pela falta de documentação precisa sobre aprendizagem a partir de bases de dados. Foi escolhida a API do BNPC para a integração a um ambiente de desenvolvimento, porque embora seu material de referência tenha sido pouco, foi o que melhor detalhou a seqüência de parâmetros a serem passados para as suas funções.

Durante o estudo da API foi relatado todo o processo de utilização da mesma com o ambiente de desenvolvimento *Visual Basic*, desde a maneira como devem ser chamadas as suas funções, bem como os parâmetros referentes a base de dados.

Uma interface foi construída para que o usuário possa realizar o processo de aquisição de conhecimento a partir da base de dados e, assim gerar a RB, que pode ser visualizada no Netica.

O *VisionBayes* foi submetido a testes utilizando-se a base de dados sobre a prevalência do Diabetes Mellitus tipo dois em sete bairros de Criciúma, e obteve uma RB, comprovando o funcionamento do protótipo desenvolvido, visando facilitar a aquisição de conhecimento em SEP.

Como trabalhos futuros sugerem-se:

a) estudar a possibilidade de integração da API do BNPC a outros ambientes de desenvolvimento;
b) estudar os mecanismos de acesso as bases de dados suportados pela API, além do DAO, utilizando a tecnologia Java via JNDI, *hipernate* ou outra;
c) utilizar a API do BNPC para a construção de uma *shell* de mineração de dados em RB para a extração de conhecimento;
d) construir um SEP por meio da aquisição de conhecimento automatizada realizada pelos recursos da API do BNPC ou de outra *shell* de mineração de dados em RB;
e) explorar os recursos de API das outras *shells* de mineração de dados em RB, podendo ser realizadas comparações entre a RB gerada pelo algoritmo da API do BNPC com a rede construída pelas API das outras *shells*, a partir de uma mesma base de dados.

REFERÊNCIAS

ACID, Silvia; CAMPOS, Luis M. ***Benedict:*** *an algorithm for learning probabilistic belief networks. Granada*, 1996. Disponível em: <http://decsai.ugr.es/gte/tr.html>. Acesso em: 10 nov. 2005.

AURELIO, Marco; VELLASCO, Marley; LOPES, Carlos Henrique. **Descoberta de Conhecimento e Mineração de Dados. Laboratório de Inteligência Computacional Aplicada –** Departamento de Engenharia Elétrica, PUC–Rio, 1999. Disponível em: <http://www.ica.ele.puc-rio.br/cursos/download/DM-apostila1.pdf>. Acesso em 16 nov. 2005.

BARONE, Dante A. C. **Sociedades artificiais:** a nova fronteira da inteligência nas máquinas. Porto Alegre: Bookman, 2003.

BARRETO, Jorge M. **Inteligência Artificial no limiar do século XXI:** Abordagem Híbrida, Simbólica, Conexionista e Evolutiva. 3. ed. Florianópolis: O autor, 2001.

BAYESWARE LIMITED. Disponível em: <http://www.bayesware.com>. Acesso em: 10 out. 2005.

BITTENCOURT, Guilherme. **Inteligência artificial:** ferramentas e teorias. Florianópolis: UFSC, 1998.

CARNEIRO, Alexandre Lênin. **Aprendizado automático em redes *bayesianas*.** 1999. 92 f. Dissertação (Mestrado em Ciência da Computação) – Universidade de Brasília, Brasília, 1999. Disponível em: <www.cic.unb.br/pg/mestrado/teses/rd-carneiro-9772847.html>. Acesso em: 2 nov. 2005.

CARVALHO, Luís Alfredo Vidal de. ***Datamining*:** A Mineração de Dados no Marketing, Medicina, Economia, Engenharia e Administração. 2. ed. São Paulo: Érica Ltda, 2002.

CHENG, Jie. *Belief network PowerConstructor.* Disponível em: <http://www.cs.ualberta.ca/~jcheng/bnpc.htm>. Acesso em: 2 fev. 2006.

CHENG, Jie; BELL, David A.; LIU, Weiru. ***Learning belief networks from data:*** *An information theory based approach. Technical Report, University of Ulster at Jordanstown,*1998. Disponível em: <http://www.cs.ualberta.ca/~jcheng/Doc/report98.pdf>. Acesso em: 10 nov. 2005.

CZEPIELEWSKI, Mauro Antonio. **ABC da saúde:** Diabetes Mellitus, 2003. Disponível em: <http://www.abcdasaude.com.br/>. Acesso em: 10 mar. 2006.

DIAS, Maria Madalena, et al. **Um modelo de formalização do processo de desenvolvimento de sistemas de descoberta de conhecimento em banco de dados.** 2001. 212 f. Tese de Doutorado (Programa de Pós-Graduação em Engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis, 2001. Disponível em: <http://teses.eps.ufsc.br/defesa/pdf/3079.pdf>. Acesso em: 17 nov. 2005.

FAYYAD, U.M.; PIATETSKY-SHAPIRO, G. & SMYTH, P. ***Knowledge Discovery and Data Mining:*** Towards a Unifying Framework. *Second International Conference on KD & DM*. Portland, Oregon, 1996.

FERNANDES, Anita Maria da Rocha. **Inteligência Artificial:** Noções Gerais. Florianópolis: Visual Books, 2003.

FILHO, Fadlo Fraige. **Diabetes mellitus tipo 2:** do diagnostico ao tratamento. 2001. Disponível em: <http://www.anad.org.br/Diabetes%20Mellitus%20Tipo%202.htm>. Acesso em: 10 mar. 2006.

GENIE, Disponível em <http://genie.sis.pitt.edu>. Acesso em: 2 fev. 2006.

HUGIN EXPERT A/S. Disponível em: <http://www.hugin.com>. Acesso em: 10 out. 2005.

KOEHLER, Cristiane. **Uma Abordagem Probabilística para Sistemas Especialistas.** 1998. 97 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, 1998.

KOEHLER, Cristiane. **Modelagem de Redes *Bayesianas* a partir da Identificação de Padrões em Base de Dados.** 2002. 44 f. Tese de Doutorado (Programa de Pós-Graduação em Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.

KOEHLER, Cristiane; NASSAR, Sílvia Modesto. **Modelagem de Redes *Bayesianas* a partir de Base de Dados Médicas**. In: SIMPOSIO DE INFORMÁTICA EN SALUD, 2002. Disponível em: <http://www.sis.org.ar/sis2002/paperssis/SIS01.pdf>. Acesso em: 17 out. 2005.

KOEHLER, Cristiane et al. **Extração Automática de Conhecimento a partir de Base de Dados:** Proposta de Algoritmo utilizando Abordagem Estatística. In: SIMPOSIO DE INFORMÁTICA EN SALUD, 2004. Disponível em: <http://www.sis.org.ar/sis2004/Trabajos/Extracao_Automatica_de_Conhecimento.pdf>. Acesso em: 20 abr. 2006.

LADEIRA, Marcelo; VICARI, Rosa Maria; COELHO, Helder. **Redes Bayesianas Multiagentes.** In TUTORIAL APRESENTADO NO ENCONTRO NACIONAL DE INTELIGÊNCIA ARTIFICIAL, 1999. Disponível em: <http://www.sbc.org.br/reic/edicoes/2002e1/tutoriais/RedesBayesianasMultiagentes.pdf >. Acesso em: 27 out. 2005.

LADEIRA, Marcelo et al. Disponível em: <http://www.din.uem.br/~hilliany/arquivos/trabalho%203%20IA/arq0221.pdf> Acesso em: 2 fev. 2006.

LUGER, George F. **Inteligência Artificial:** estruturas e estratégias para a solução de problemas complexos. 4 ed. Porto Alegre: Bookmann, 2004.

MANARIN, Daiane de Nez. **Aquisição de Conhecimento em Sistemas Especialistas Probabilísticos por meio da Descoberta de Conhecimento em Base de Dados para Construção de Redes Bayesianas.** 2004. 112 f. Monografia (Bacharel em Ciência da Computação) - Universidade do Extremo Sul Catarinense, Criciúma, 2004.

MELLO, Luis Cesar de. **Descoberta de conhecimento em banco de dados (mineração de dados) e redes *bayesianas***: estado da arte. 2001. 56 f. Dissertação (Mestrado em Ciência da Computação Convênio UFRGS/FACCAR-Rolância) – Curso de Ciência da Computação, Universidade Federal de Rio Grande do Sul, Porto Alegre, 2001. Disponível em: <http://www.inf.ufrgs.br/procpar/direto/trabalhos/TI-LuisCesarMello.pdf>. Acesso em: 10 out. 2005.

MILECH, Adolpho et al. **Consenço Brasileiro sobre Diabetes:** Diagnóstico e classificação do diabetes mellitus e tratamento do diabetes mellitus tipo 2. 2002. Disponível em: <www.diabetes.org.com.br>. Acesso em: 10 mar. 2006.

OFFICE, Microsoft. **Data Access Objects.** Disponível em: <http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/office97/html/web/011.asp>. Acesso em: 10 abr. de 2006.

MOTTA, Custódio Gouvêa Lopes da. **Introdução a Técnicas de *Data Mining* – DM.** Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF), Rio de Janeiro, 2005. Disponível em: <http://www.lncc.br/verao/verao05/arquivos/MiniCursoDMLNCC050202C.pdf>. Acesso em 22 out. 2005.

NASSAR, Sílvia Modesto. **Tratamento de Incerteza:** Sistemas Especialistas Probabilísticos. 2005. Disponível em: <http://www.inf.ufsc.br/~silvia/disciplinas/sep/MaterialDidatico.pdf>. Acesso em: 10 out. 2005.

NETO, Antonio Lopes de Sá; LIMA, Izaias M.; AFONSO, Mônica B. **Aplicação de Redes *Bayesianas* para Extração de Conhecimento de Bases de Dados**. 2002. 56 f. Trabalho de Conclusão de Curso (Curso de Ciência da Computação) - Universidade da Amazônia, Belém, 2002. Disponível em: <http://www.cci.unama.br/margalho/portaltcc/tcc2002/Redes*Bayesianas*.PDF>. Acesso em: 20 out. 2005.

ORTIZ, Maria Carolina Alves; ZANETTI, Maria Lúcia**. Levantamento dos fatores de risco para diabetes mellitus tipo 2 em uma instituição de ensino superior**. 2001. Disponível em: <www.scielo.br/pdf/rlae/v9n3/11499.pdf >. Acesso em: 10 mar. 2006.

PASINI, Hamilton. **SEPE: Sistema Especialista Probabilístico para Apoio ao Diagnóstico de Potencial Econômico.** 2002. 67 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2002.

PASSOS, Emmanuel; GOLDSCHIMIDT, Ronaldo. **Data Mining:** um guia prático. Rio de Janeiro: Elsevier, 2005.

PEARL, Judea. **Probabilistic Reasoning in Intelligent Systems:** networks of plausible inference. California, San Mateo: Morgan Kaufmann Publishers, 1988.

RABUSKE, Antônio Renato. **Inteligência Artificial.** Florianópolis: UFSC, 1995.

REZENDE, Solange Oliveira**. Sistemas Inteligentes:** fundamentos e aplicações. São Paulo: Manole, 2003.

ROMÃO, Wesley. **Descoberta de Conhecimento Relevante em Banco de Dados sobre Ciência e Tecnologia.** 2002. 238 f. Tese de Doutorado (Programa de Pós-Graduação em Engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis, 2002. Disponível em: <http://teses.eps.ufsc.br/defesa/pdf/3469.pdf>. Acesso em: 10 out. 2005.

ROVARIS NETO, Eugênio. **E-Bayes: Redes Bayesianas para Diagnóstico da Evasão Escolar no Ensino Superior**. 2002. 75 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2002.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial.** Tradução da 2ª edição. Rio de Janeiro: Campus, 2004.

SILVA, Wagner Teixeira da; LADEIRA, Marcelo. **Mineração de Dados em Redes Bayesianas.** Disponível em: <http://www.din.uem.br/~williany/arquivos/trabalho%203%20IA/jai02cap6.pdf>. Acesso: em 20 abr. 2006.

BIBLIOGRAFIA RECOMENDADA

BOUCKAERT, R. R. **Bayesian belief networks:** from inference to construction. PhD thesis, Faculteit Wiskunde en Informatica, Utrech Universiteit, June 1995.

HECKERMAN, David. ***A tutorial on learning Bayesian networks***. *Technical Report, Microsoft Research, Advanced Technology Division, Microsoft Corporation*, 1995. Disponível em: <ftp://ftp.research.microsoft.com/pub/tr/tr-95-06.pdf >. Acesso em: 1 dez. 2005.

HERSKOVITS, Edward. ***Computer - based probabilistic - network construction.*** 1991. 225 f. (*Mastership in philosophy) – University of Pittsburgh*, 1991. Disponível em: http://www.herskovits.org/ehh/publications/Thesis.pdf >. Acesso em: 1 dez. 2005.

APÊNDICE A – O ALGORITMO CBL

O algoritmo implementado na *shell* BNPC e, conseqüentemente em sua API, é o CBL, que utiliza o método de análise de dependências para a construção de RB a partir de bases de dados, assumindo que seus campos não possuam valores ausentes (CHENG, 2006), (CARNEIRO, 1999).

O CBL analisa as relações de dependência entre as variáveis escolhidas da base de dados utilizando algum teste de independência condicional. Estes testes com grandes conjuntos de condições podem apresentar problemas a menos que o volume de informações da base de dados seja grande (CARNEIRO, 1999), (KOEHLER et al, 2004).

O algoritmo, cujos autores são Cheng, Bell e Liu (1998), apresenta duas versões denominadas A e B, sendo esta última uma extensão da primeira. A diferença entre as duas é que a versão A considera somente bases de dados sem a presença de valores ausentes, e a versão B, permite a presença de variáveis com valores não definidos (CARNEIRO, 1999).

Seu funcionamento baseia-se no recebimento, como entrada, de uma base de dados e da ordenação total de suas variáveis (que serão os nós da rede) na versão A, já que na versão B, não há a necessidade da ordenação delas. Como saída, o algoritmo constrói a RB (CARNEIRO, 1999), (SILVA; LADEIRA, 2006).

O CBL possui três fases: *drafting*, *thickening* e *thinning*. A primeira delas é essencialmente o algoritmo de construção da árvore de Chow e Liu's; as duas outras fases são realizadas para estender a construção da árvore à construção de uma RB. O *drafting* calcula a informação mútua de cada par dos nós, a fim de criar um esboço, representado por um grafo conectado sem laços, com tal informação. Na fase de

thickening, adiciona-se bordas quando os pares dos nós não podem ser *d-separation*[23], tendo-se como resultado final desta etapa, uma estrutura de uma mapa de independência (*I-map*) do modelo de dependência. No *thinning*, cada arco do *I-map* é examinado utilizando os testes de independência condicional, removendo-o caso os dois nós do arco possam ser *d-separation*. O algoritmo, nesta fase, também realiza um procedimento a fim de orientar os arcos do grafo, tendo-se como resultado final, um *I-map* mínimo (KOEHLER et al, 2004).

Nos testes realizados pelos autores na rede ALARM[24], foi possível verificar que a versão A aprendeu a estrutura original da rede com apenas um arco ausente (em relação à rede ideal) em menos de quinze minutos, sendo que a maior parte desse processamento, concentrou-se na primeira fase do algoritmo, que consumiu dez minutos e sete segundos. Utilizando a versão B do CBL com a mesma rede e nas mesmas condições que foi realizado o teste anterior, obteve-se uma RB com dois arcos a menos do que a rede original em aproximadamente dezesseis minutos.

Dessa forma, o CBL pode ser considerado um dos algoritmos mais eficientes e ágeis para aprendizagem de RB a partir de bases de dados (KOEHLER et al, 2004), sendo encontrado também no UnBBayes, além da *shell* BNPC, dos autores do algoritmo.

[23] Para quaisquer três conjuntos de nós disjuntos X, Y e Z em uma RB, X é *d-separation* de Y por Z, se não existir caminho direcionado ativo entre X e Y.
[24] RB utilizada em testes, que apresenta a monitoração dos pacientes anestesiados

ANEXO A – PROTOCOLO SOBRE O DIABETES MELLITUS TIPO DOIS

Unidade Básica de Saúde:				Número do Prontuário
IDENTIFICAÇÃO DO USUÁRIO				
Nome	Idade (anos)	Data Nascimento	Sexo M F	
DADOS CLÍNICOS DO PACIENTE				
Pressão Arterial Sistólica	Pressão Arterial Diastólica	Altura	Peso (kg)	IMC *
Glicemia de jejum	Hemoglobina glicosilada	Glicemia pós- prandial	Glicemia capilar (hemoglicoteste)	

Fatores de risco e Doenças concomitantes	Não	Sim	Presença de Complicações	Não	Sim
Antecedentes Familiares - cardiovasculares			Infarto Agudo Miocárdio		
Hipertensão Artéria **I			Outras coronariopatias		
Tabagismo			AVC		
Sedentarismo			Pé diabético		
Sobrepeso/Obesidade			Amputação por diabetes		
Diabetes gestacional			Doença Renal		
Outro (a):					

TRATAMENTO

Não Medicamentoso:

	Medicamentoso							Unidades/dia
	Comprimidos/dia							
Tipo	**1/2**	**1**	**2**	**3**	**4**	**5**	**6**	
Hidroclorotiazida 25mg								
Propranolol 40mg								
Captopril 25mg								
Glibenclamida 5mg								**Insulina**
Metformina 850 mg								

Outros SIM NÃO

Outros dados demográficos

Raça/Cor	
código	Descrição
	Branca
	Preta
	Amarela
	Parda
	Indígena

Situação familiar/Conjugal	
Código	Descrição
1	Convive c/ companheira(o) e filho(s)
2	Convive c/ companheira(o) c/ laços conjugais e s/ filhos
3	Convive c/ companheira(o), filhos e/ou outros familiares
4	Convive c/ familiares, com companheira(o)
5	Convive c/ outra(s) pessoa(s), sem laços consangüíneos e/ou laços conjugais
5	Vive só

Escolaridade	
Código	Descrição
01	Não sabe ler/escrever
02	Alfabetizado
03	Fundamental incompleto (1º grau incompleto)
04	Fundamental completo (1º grau completo)
05	Médio Incompleto (2º grau incompleto)
06	Médio completo (2º grau completo)
07	Superior incompleto
08	Superior completo
09	Especialização/Residência
10	Mestrado
11	Doutorado

***Sobrepeso ou Obesidade – classificação de acordo com a tabela:**

Classificação	IMC (peso em Kg/altura ao quadrado)
Normal	18,5-24,9
Sobrepeso	25,0-29,9
Obeso Classe I	30,0-34,9
Obeso Classe II	35,0-39,9
Obeso Classe III	>/= 40,0

****Classificação da Pressão Arterial (mmHg)**

Normal alta PAS 135-139 ou PAD 85-89	Grau 1 Hipertensão leve PAS 140-159 ou PAD 90-99	Grau 2 Hipertensão moderada PAS 160-179 ou PAD 100-109	Grau 3 Hipertensão grave PAS $\geq$ 180 ou PAD $\geq$ 110