

Universidade Federal do Rio de Janeiro

APRENDIZADO INDUTIVO DE REDES BAYESIANAS:  
ALÉM DA PRECISÃO NA TAREFA DE CLASSIFICAÇÃO

Edimilson Batista dos Santos

Edimilson Batista dos Santos

D.Sc.  
COPPE/UFRJ  
2011

2011

## APRENDIZADO INDUTIVO DE REDES BAYESIANAS: ALÉM DA PRECISÃO NA TAREFA DE CLASSIFICAÇÃO

Edimilson Batista dos Santos

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Civil, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Civil.

Orientadores: Nelson Francisco Favilla Ebecken  
Estevam Rafael Hruschka Júnior

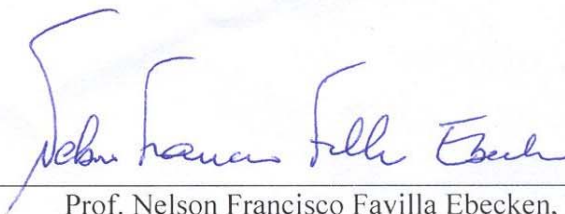
Rio de Janeiro  
Dezembro de 2011

APRENDIZADO INDUTIVO DE REDES BAYESIANAS: ALÉM DA PRECISÃO  
NA TAREFA DE CLASSIFICAÇÃO

Edimilson Batista dos Santos

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ  
COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA  
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS  
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM  
CIÊNCIAS EM ENGENHARIA CIVIL.

Examinada por:



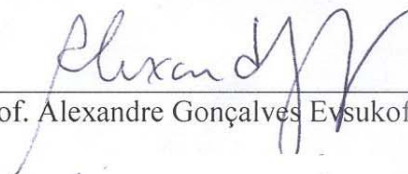
Prof. Nelson Francisco Favilla Ebecken, D.Sc.



Prof. Estevam Rafael Hruschka Júnior, D.Sc.



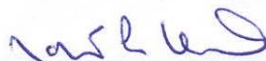
Profª Beatriz de Souza Leite Pires de Lima, D.Sc.



Prof. Alexandre Gonçalves Eysukoff, Dr.



Prof. Helio Jose Correa Barbosa, D.Sc.



Profª Marta Lima de Queiroz Mattoso, D.Sc.

RIO DE JANEIRO, RJ – BRASIL  
DEZEMBRO DE 2011

Santos, Edimilson Batista

Aprendizado Indutivo de Redes Bayesianas: além da Precisão na Tarefa de Classificação/Edimilson Batista dos Santos. – Rio de Janeiro: UFRJ/COPPE, 2011.

XII, 104 p.: il.; 29,7 cm.

Orientador: Nelson Francisco Favilla Ebecken

Estevam Rafael Hruschka Junior

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Civil, 2011.

Referências Bibliográficas: p. 96-104.

1. Redes Bayesianas. 2. Algoritmos Evolucionários. 3. Classificação de Dados. I. Ebecken, Nelson Francisco Favilla et al. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Civil. III. Título.

À minha família:  
Valquíria,  
Geraldo, Maria dos Anjos,  
Micaela, Roberto e José Marcos Júnior.

## **Agradecimentos**

Agradeço a Deus pela saúde e disposição necessárias para conclusão deste trabalho.

Aos meus orientadores, Nelson Francisco Favilla Ebecken e Estevam Rafael Hruschka Junior, por toda confiança, apoio, auxílio, motivação e entusiasmo, fazendo-me trabalhar com dedicação e prazer, mesmo diante das dificuldades.

À Valquíria, que me incentivou desde o início e suportou a minha ausência, com paciência e compreensão. E a sua família, que sempre tiveram o maior carinho comigo.

Aos meus pais e irmãos, que sempre me apoiaram.

Aos meus amigos, pelas contribuições e pela companhia.

Ao professor Alexandre G. Evsukoff, por todas as contribuições dadas na banca de qualificação deste trabalho.

Aos professores do Departamento de Engenharia Civil da COPPE/UFRJ e da UFSCar, que contribuíram para o meu conhecimento durante este período.

Ao NACAD (Núcleo de Atendimento a Computação de Alto Desempenho) – COPPE/UFRJ, pela execução de experimentos realizados neste trabalho.

Aos revisores dos periódicos e conferências, que conheceram partes isoladas da tese através de artigos submetidos e trouxeram contribuições relevantes.

Aos funcionários do Departamento de Computação da UFSCar, que permitiram meu acesso ao laboratório.

Aos funcionários da COPPE, em especial à Egna, por todo apoio prestado.

A todos, que de forma direta ou indireta, contribuíram para a realização deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

## APRENDIZADO INDUTIVO DE REDES BAYESIANAS: ALÉM DA PRECISÃO NA TAREFA DE CLASSIFICAÇÃO

Edimilson Batista dos Santos

Dezembro/2011

Orientadores: Nelson Francisco Favilla Ebecken

Estevam Rafael Hruschka Júnior

Programa: Engenharia Civil

O desenvolvimento de métodos automáticos para aprender estruturas de uma Rede *Bayesiana* (RB) diretamente a partir de dados é um problema relevante e considerado uma tarefa difícil, porque o número de possíveis estruturas cresce exponencialmente de acordo com o número de variáveis. Geralmente, para reduzir o espaço de busca, algumas restrições podem ser impostas durante o processo de indução da RB. Uma restrição possível é a definição de uma ordenação das variáveis. Definir uma ordenação adequada das variáveis é, contudo, um problema complexo a ser executado, principalmente porque requer conhecimento prévio sobre o domínio. Trabalhos anteriores, na literatura, sugerem o uso de Algoritmos Evolucionários para encontrar uma ordenação de variáveis adequada ao aprendizado de estruturas de redes *Bayesianas*. No entanto, algoritmos evolucionários podem ser computacionalmente custosos, assim, o uso de operadores genéticos específicos para o problema da ordenação de variáveis pode torná-los mais eficientes. Para os casos onde a busca por uma ordenação de variáveis é adequada, este trabalho apresenta dois novos operadores genéticos e um novo algoritmo adaptativo híbrido que busca tal ordenação para otimizar o aprendizado das redes. Para os casos onde uma ordenação não é indicada, são definidos um método de indução de classificadores *Bayesianos*, chamado DMBC, e uma versão aproximada, chamada A-DMBC. Os resultados experimentais mostram que os métodos propostos são consistentes e promissores.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

BAYESIAN NETWORKS INDUCTIVE LEARNING: BEYOND THE PRECISION  
IN THE CLASSIFICATION TASK

Edimilson Batista dos Santos

Dezembro/2011

Advisors: Nelson Francisco Favilla Ebecken

Estevam Rafael Hruschka Júnior

Department: Civil Engineering

The development of automatic methods for learning Bayesian network structures directly from data is a relevant problem, and is considered a difficult task because the number of possible structures grows exponentially according to the number of variables. In order to reduce this search space, some constraints can be imposed during the induction process. One possible constraint is the definition of a variable ordering. Defining a suitable variable ordering is, however, a complex problem to be performed mainly because it requires prior knowledge about the model. Previous works in the literature suggest that the use of evolutionary algorithms for dealing with variable ordering, when learning a Bayesian network structure from data, is worth pursuing. However, evolutionary algorithms may be computationally costly and the use of specific genetic operators for variable ordering problem can make them more efficient. To the cases where a variable ordering is suitable, this work presents two new genetic operators and a new hybrid adaptive algorithm which look for such a ordering for optimizing the network learning. To the cases where a variable ordering is not indicated, a method of induction of Bayesian classifiers named DMBC and an approximate version named A-DMBC are defined. The experimental results show that the proposed methods are consistent and promising.



# Sumário

1	Introdução .....	1
1.1	Considerações Iniciais .....	1
1.2	Relevância .....	2
1.3	Objetivos e Contribuições .....	3
1.4	Organização do Documento .....	4
2	Redes <i>Bayesianas</i> .....	5
2.1	Definições .....	5
2.2	Inferência <i>Bayesiana</i> .....	6
2.3	O Conceito de Cobertura de <i>Markov</i> .....	10
2.4	Aprendizado de Redes <i>Bayesianas</i> a partir de Dados .....	12
2.4.1	Métodos com base em Busca Heurística .....	13
2.4.2	Métodos com base em Independência Condicional .....	15
2.5	Aprendizado de Classificadores <i>Bayesianos</i> a partir de Dados .....	18
2.5.1	TAN .....	19
2.6	O Problema do Aprendizado de Estruturas .....	22
2.6.1	A Ordenação de Variáveis .....	23
2.7	Considerações Finais .....	25
3	Algoritmos Evolucionários .....	26
3.1	Definições .....	26
3.2	Representação do Problema .....	28
3.3	Métodos de Seleção .....	29
3.4	Operadores Genéticos Clássicos .....	30
3.5	Operadores Genéticos de Permutação .....	30
3.6	Parâmetros Genéticos .....	31
3.7	MOS: <i>Multiple Offspring Sampling</i> .....	31
4	O Problema da Ordenação de Variáveis .....	34
4.1	Considerações Iniciais .....	34
4.2	Influência da Ordenação de Variáveis .....	35
4.2.1	Valor da Função g .....	36
4.2.2	Taxa de Classificação .....	38
4.2.3	Estimativas de Probabilidades .....	39
4.3	Busca da Ordenação usando Estratégias Evolucionárias .....	40

4.4	Considerações Finais .....	41
5	Aprendizado de Estruturas com DMBC e A-DMBC .....	43
5.1	DMBC: <i>Dynamic Markov Blanket Classifier</i> .....	43
5.2	A-DMBC: <i>Approximate Dynamic Markov Blanket Classifier</i> .....	45
5.3	Avaliação Experimental .....	47
5.3.1	Configuração Experimental .....	47
5.3.2	Experimentos e Análises de Resultados .....	50
5.3.3	Discussão .....	56
6	Métodos Evolucionários de Busca da Ordenação .....	58
6.1	Considerações Iniciais .....	58
6.2	Métricas de Pontuação utilizadas como Funções de Aptidão .....	59
6.2.1	A Métrica de Entropia .....	60
6.2.2	A Métrica AIC .....	60
6.2.3	A Métrica BDe .....	60
6.2.4	A Métrica MDL .....	61
6.2.5	Experimentos e Análises de Resultados .....	61
6.2.6	Discussão .....	65
6.3	Operador de Mutação com base em Distância (DMO) .....	66
6.3.1	Experimentos e Análises de Resultados .....	68
6.3.2	Discussão .....	72
6.4	Operador de Cruzamento de Múltiplos Pontos Aleatórios (RMX) .....	73
6.4.1	Experimentos e Análises de Resultados .....	75
6.4.2	Comparação entre RMX e DMO.....	79
6.4.3	Discussão .....	81
6.5	Algoritmo VOMOS .....	81
6.5.1	Experimentos e Análises de Resultados .....	84
6.5.1.1	Comparação entre VOMOS e Algoritmos Genéticos .....	85
6.5.1.2	Comparação entre VOMOS e VOEA-RMX .....	87
6.5.2	Discussão .....	89
6.6	Uma Aplicação Prática .....	89
6.7	Considerações Finais .....	91
7	Conclusões e Trabalhos Futuros.....	92
7.1	Avaliação dos Métodos DMBC e A-DMBC.....	93
7.2	Avaliação das Estratégias Evolucionárias .....	94

7.3	Trabalhos Futuros .....	95
8	Referências .....	96

# Lista de Figuras

Figura 2.1. Estrutura de Rede <i>Bayesiana</i> . ....	6
Figura 2.2. Rede <i>Bayesiana</i> representando a distribuição de probabilidade discutida no Exemplo 2.1.....	8
Figura 2.3. Um fragmento de uma poliárvore. ....	10
Figura 2.4. Os nós hachurados representam a Cobertura de <i>Markov</i> do nó A. ....	11
Figura 2.5. Pseudocódigo do algoritmo K2, adaptado de [16]. ....	15
Figura 2.6. GDA onde o conjunto de nós {C, D} é d-separado do nó F pelo nó E.....	17
Figura 2.7. GDA onde o nó C é d-separado do nó F pelos nós {E, D}.....	17
Figura 2.8. Pseudocódigo do algoritmo PC [5]. ....	18
Figura 2.9. Comparação entre as estruturas dos modelos NB e TAN. ....	20
Figura 2.10. Fases do aprendizado de um modelo TAN. ....	22
Figura 3.1. Pseudocódigo de um típico algoritmo evolucionário (adaptado de [54]). ....	27
Figura 3.2. Pseudocódigo do algoritmo genético. ....	28
Figura 3.3. Pseudocódigo do algoritmo MOS [7]. ....	33
Figura 4.1. Estruturas originais das redes <i>Bayesianas Asia</i> e Câncer Metastático [60]....	37
Figura 4.2. Estruturas das redes <i>Bayesianas Asia</i> , aprendidas pelo K2, com ordenação ótima (a), ordenação intermediária (b) e ordenação ruim (c). ....	38
Figura 4.3. Estruturas das redes <i>Bayesianas</i> do Câncer Metastático, aprendidas pelo K2, com ordenação ótima (a), ordenação intermediária (b) e ordenação ruim (c)....	38
Figura 5.1. Pseudocódigo do algoritmo DMBC. ....	45
Figura 5.2. Pseudocódigo do algoritmo A-DMBC.....	46
Figura 5.3. Estruturas de redes <i>Bayesianas</i> para o primeiro cenário experimental.....	48
Figura 5.4. Estruturas de redes <i>Bayesianas</i> para o segundo cenário experimental. ....	49
Figura 5.5. Estruturas de redes <i>Bayesianas</i> para o terceiro cenário experimental. ....	50
Figura 5.6. Número de avaliações da função g. ....	52
Figura 5.7. Taxas médias de classificação correta em uma estratégia de validação cruzada <i>10-fold</i> . ....	53
Figura 5.8. Distância euclidiana entre os valores de probabilidade verdadeiros e aqueles estimados pelos algoritmos.....	54
Figura 5.9. Divergência <i>Kullback-Leibler</i> entre os valores de probabilidade verdadeiros e aqueles estimados pelos algoritmos. ....	55

Figura 6.1. Redes <i>Bayesianas</i> representando os domínios <i>Synthetic 1</i> , <i>Synthetic 2</i> e <i>Synthetic 3</i> . As representações gráficas foram criadas usando o software GeNie [60].	62
Figura 6.2. Redes <i>Bayesianas</i> representando os domínios <i>Synthetic 50-50</i> , <i>Synthetic 50-100</i> e <i>Synthetic 50-200</i> . As representações gráficas foram criadas usando o software <i>Weka</i> [81].	63
Figura 6.3. Algoritmo DMO.	66
Figura 6.4. Rede <i>Bayesiana</i> representando o domínios <i>Synthetic 50-300</i> . A representação gráfica foi criada usando o software <i>Weka</i> [81].	69
Figura 6.5. Estrutura de uma rede <i>Bayesiana</i> hipotética.	72
Figura 6.6. Algoritmo RMX.	74
Figura 6.7. Aplicação do operador RMX.	75
Figura 6.8. Redes <i>Bayesianas</i> representando os domínios da Tabela 6.8. As representações gráficas foram criadas usando o software <i>Weka</i> [81].	77
Figura 6.9. Pseudocódigo do algoritmo VOMOS.	82
Figura 6.10. Fluxograma do algoritmo VOMOS_1.	83
Figura 6.11. Fluxograma do algoritmo VOMOS_N.	84
Figura 6.12. Estrutura do classificador DMBC para o TEP.	90
Figura 6.13. Estrutura do classificador VOMOS_1 para o TEP.	91

## Lista de Tabelas

Tabela 4.1. Descrição dos conjuntos de dados, constando o nome do domínio (DOMÍNIO), o número de atributos (AT), o número de ordenações possíveis (OVP), o número aproximado (por conta do arredondamento) de ordenações ótimas (OVO) e o número aproximado (por conta do arredondamento) de ordenações não ótimas (OVNO). ....	35
Tabela 4.2. Valores de g obtidos por K2, com ordenação ótima, intermediária e ruim. ....	37
Tabela 4.3. Ordenações ótima, intermediária e ruim, fornecidas ao K2 para a indução das redes. ....	37
Tabela 4.4. Taxas de classificação obtidas por K2, com ordenação ótima, intermediária e ruim. ....	39
Tabela 4.5. Distância entre os valores das probabilidades verdadeiras e os valores das probabilidades estimadas pelo K2, com ordenação ótima, intermediária e ruim. ....	40
Tabela 5.1. Conjuntos de dados usados nos experimentos: número de atributos (AT), número de instâncias (IN) e número de valores da classe (NV). ....	50
Tabela 5.2. Número de chamadas à função g. ....	51
Tabela 5.3. Taxas médias de classificação correta (TMCC) obtidas por validação cruzada <i>10-fold</i> . ....	53
Tabela 5.4. Distância Euclidiana entre os valores de probabilidades verdadeiros e os valores de probabilidades estimados pelos algoritmos. ....	54
Tabela 5.5. Divergência KL entre os valores de probabilidades verdadeiros e os valores de probabilidades estimados pelos algoritmos. ....	55
Tabela 6.1. Descrição dos conjuntos de dados, constando o nome do conjunto, número de atributos mais a classe (AT), número de instâncias (IN) e número de valores da classe (NV). ....	62
Tabela 6.2. Número de arcos extras (+), número de arcos ausentes (-) e número de arcos invertidos (R). ....	64
Tabela 6.3. Número de gerações necessárias à convergência. ....	65
Tabela 6.4. Porcentagens de classificação obtidas pelas versões de VOGA. ....	65
Tabela 6.5. Descrição dos conjuntos de dados, constando o nome do conjunto, número de atributos mais a classe (AT), número de instâncias (IN) e número de valores da classe (NV). ....	68

Tabela 6.6. Scores Bayesianos (função g) obtidos por VOGA e pelas versões de VOGA. ....	71
Tabela 6.7. Número de gerações necessárias para a convergência dos algoritmos VOGA e VOGA . ....	71
Tabela 6.8. Descrição dos conjuntos de dados, contendo o nome do conjunto de dados (domínio), número de atributos mais a classe (AT), número de instâncias (IN) e número de arcos (AR). ....	76
Tabela 6.9. Scores Bayesianos (função g). ....	78
Tabela 6.10. Número de gerações necessárias para convergência. ....	78
Tabela 6.11. Scores Bayesianos (função g). ....	79
Tabela 6.12. Número de gerações necessárias para convergência. ....	80
Tabela 6.13. Principais diferenças entre as duas versões de VOMOS. ....	85
Tabela 6.14. Média dos scores Bayesianos (Função g) e número médio de gerações antes da convergência, respectivamente, obtidos pelos algoritmos genéticos com população de tamanho 10, combinando os operadores genéticos. ....	86
Tabela 6.15. Média dos scores Bayesianos (Função g) e número médio de gerações antes da convergência, respectivamente, obtidos pelos algoritmos genéticos com população de tamanho 50, combinando os operadores genéticos. ....	86
Tabela 6.16. Média dos scores Bayesianos (Função g) e número médio de gerações antes da convergência, respectivamente, obtidos por VOMOS_1 e VOMOS_N, com população de tamanho 10 e 50. ....	86
Tabela 6.17. Média dos scores Bayesianos (Função g). ....	87
Tabela 6.18. Número médio de gerações para a convergência. ....	88
Tabela 6.19. Taxas de classificação corretas. ....	90

# 1 Introdução

## 1.1 Considerações Iniciais

Redes *Bayesianas*, também conhecidas como redes de crença ou redes causais, são modelos gráficos probabilísticos utilizados para raciocínio baseado em incerteza, onde os nós representam as variáveis e os arcos representam a ligação direta entre elas. Nas duas últimas décadas, redes *Bayesianas* [1] tornaram-se um método muito utilizado para representação de conhecimento e raciocínio envolvendo incerteza.

Uma rede *Bayesiana* é formada basicamente por dois componentes: a estrutura gráfica (grafo direcionado acíclico – GDA) e os parâmetros numéricos (tabelas de probabilidades condicionais). Estes dois componentes podem ser aprendidos indutivamente a partir de dados. Primeiro, deve-se induzir a estrutura e, a seguir, com a estrutura conhecida, se aprende os parâmetros numéricos. Se a estrutura já é conhecida, então o problema se restringe a aprender os parâmetros numéricos. Enquanto o aprendizado de parâmetros numéricos é relativamente simples (quando a estrutura já é conhecida), o aprendizado da estrutura é um assunto, em geral, complexo e ainda não possui solução exata.

As redes *Bayesianas* podem ser usadas tanto em tarefas de aprendizado supervisionado quanto em tarefas de aprendizado não supervisionado. Quando uma tarefa de aprendizado supervisionado é conduzida, a rede *Bayesiana* é geralmente chamada de Classificador *Bayesiano*.

A ordenação das variáveis de um domínio pode ter uma importante função no processo de aprendizado de redes *Bayesianas* (e classificadores *Bayesianos*) auxiliando a reduzir o espaço de busca deste problema. Nesta tese é apresentado um estudo mais aprofundado sobre a influência da ordenação de variáveis no aprendizado de redes *Bayesianas* e são propostos métodos e algoritmos que permitem otimizar a indução de redes *Bayesianas* a partir de dados.



## 1.2 Relevância

O aprendizado indutivo de redes *Bayesianas* se tornou uma área de pesquisa bastante ativa nos últimos anos. Bons resultados alcançados ultimamente motivaram o desenvolvimento de muitos algoritmos de aprendizado de redes *Bayesianas* [2].

O espaço de busca para uma rede com  $n$  variáveis tem dimensão exponencial. Assim, encontrar a estrutura de rede que melhor represente as dependências entre as variáveis é considerado um problema NP-Completo [3], sendo difícil identificar a melhor solução para todos os problemas de aplicação [4]. Neste sentido, modelos aproximados podem ser induzidos e o espaço de busca deste processo provavelmente será reduzido. Para isto, algumas restrições são geralmente impostas e os algoritmos frequentemente obtêm bons resultados, com esforço computacional aceitável. Uma restrição muito comum nos algoritmos de aprendizado de redes *Bayesianas* é a ordenação prévia das variáveis utilizadas na definição do problema [5].

A ordenação das variáveis pode representar uma função importante no processo de indução de redes *Bayesianas*, visto que há algoritmos de aprendizado que dependem desta ordenação para determinar a direção dos arcos da estrutura da rede. Mesmo em algoritmos que não exigem uma ordenação prévia das variáveis, esta ordem pode auxiliar na otimização do aprendizado. É importante ressaltar que, da mesma forma que uma boa ordenação das variáveis pode melhorar os resultados da geração de uma rede *Bayesiana*, uma ordem que não reflita de forma correta o relacionamento das variáveis, pode inserir erros no processo de aprendizagem.

A ordenação das variáveis é um tema bastante debatido, principalmente no contexto de modelagem causal. Quando se deseja criar uma representação causal do relacionamento das variáveis de um problema, uma das maiores dificuldades está na definição de qual variável é a causa e qual é a consequência em um relacionamento qualquer. Vários estudos já foram realizados sobre este tema, mas não se tem uma forma ideal para esta definição [6].

Uma vez que não existem métodos automáticos computacionalmente eficientes na busca de uma ordenação adequada de variáveis, o objetivo técnico desta tese é investigar, propor, implementar e avaliar métodos que identifiquem tal ordenação para otimizar o aprendizado supervisionado de redes *Bayesianas* a partir de dados e, também,

investigar e identificar situações onde a busca pela melhor ordenação das variáveis pode não ser necessária.

### 1.3 Objetivos e Contribuições

O principal objetivo desta tese pode ser dividido em duas linhas de investigação. Na primeira, pretende-se identificar situações onde a busca por uma ordenação de variáveis é ou não adequada e propor métodos e algoritmos adequados para tais casos. Na segunda linha de pesquisa, busca-se investigar, propor e implementar métodos e algoritmos, com base na computação evolutiva, para identificar uma ordenação adequada de variáveis, visando à otimização do aprendizado indutivo de redes *Bayesianas*.

Visando alcançar o seu objetivo principal, este trabalho apresenta três contribuições principais. A primeira contribuição é a definição de um método de indução de classificadores *Bayesianos* chamado DMBC (*Dynamic Markov Blanket Classifier*) e de uma versão aproximada do mesmo chamada A-DMBC (*Approximate Dynamic Markov Blanket Classifier*). Estes algoritmos foram desenvolvidos especialmente para problemas de classificação e apresentaram resultados consistentes. Uma característica inovadora dos métodos DMBC e A-DMBC é poderem ser utilizados sem a necessidade da busca de uma ordenação adequada de variáveis antes da indução do classificador. Mesmo sem impor esta restrição no processo de aprendizado, os algoritmos são capazes de contornar algumas limitações indesejáveis e, ao mesmo tempo, melhorar o tempo computacional necessário para aprender um classificador de rede *Bayesiana*.

A segunda contribuição compreende a teoria de redes *Bayesianas* e computação evolucionária, pois trata-se do desenvolvimento de dois operadores genéticos específicos para a busca da ordenação adequada à otimização do aprendizado de redes *Bayesianas*. O primeiro operador criado tem características de um operador de mutação e foi chamado de DMO (*Distance-based Mutation Operator*) e o segundo é um operador de cruzamento chamado de RMX (*Random Multi-point Crossover Operator*). Estes dois operadores demonstraram ser capazes de melhorar a qualidade da ordenação das variáveis e tenderam a melhorar o aprendizado de estruturas de redes *Bayesianas*, principalmente em domínios maiores e densamente conectados.

A terceira contribuição importante é a utilização de um algoritmo híbrido, capaz de combinar várias abordagens evolucionárias, chamado de MOS (*Multiple Offspring Sampling*) [7][8], utilizando o operador de cruzamento RMX. O algoritmo desenvolvido foi chamado de VOMOS (*Variable Ordering Multiple Offspring Sampling*) e explora o poder de diferentes operadores evolucionários para encontrar ordenações ótimas.

## 1.4 Organização do Documento

A organização desta tese está definida como se segue. Os dois próximos capítulos abordam de maneira geral os conceitos que fundamentam o desenvolvimento desta pesquisa. Em seguida, são apresentados os métodos desenvolvidos e, ao final, as conclusões e trabalhos futuros.

Mais especificamente, no Capítulo 2, são destacados os conceitos fundamentais da teoria de redes *Bayesianas*, incluindo a definição do processo de aprendizado. Como exemplos, são apresentados alguns algoritmos de aprendizado de redes *Bayesianas* e de classificadores *Bayesianos*.

O Capítulo 3 apresenta uma introdução sobre computação evolucionária e as características dos algoritmos evolucionários. É destacado o conceito de algoritmos genéticos, a forma de representação do problema e os operadores genéticos. Além disso, é apresentado uma visão geral sobre a abordagem MOS.

No Capítulo 4, destaca-se o problema da ordenação de variáveis, apresentando sua influência no processo de aprendizado de estruturas das redes e algumas abordagens encontradas na literatura, as quais aplicam métodos evolucionários para realizar a busca.

O Capítulo 5 apresenta os métodos DMBC e A-DMBC, desenvolvidos neste trabalho, que aprendem estruturas *Bayesianas* para a tarefa de classificação sem a restrição da ordenação de variáveis. Já o Capítulo 6, apresenta os operadores genéticos específicos propostos para a busca da ordenação, DMO e RMX, e o algoritmo VOMOS. Além disso, estes dois capítulos apresentam os experimentos realizados e discutem os resultados obtidos.

No Capítulo 7, são apresentados as conclusões e os trabalhos futuros.

## 2 Redes *Bayesianas*

### 2.1 Definições

Redes *Bayesianas* são representações gráficas de distribuição de probabilidades e têm sido muito utilizadas para representação de incerteza em Inteligência Artificial. Segundo [9], elas representam um papel crucial em modernos sistemas especialistas, máquinas de diagnósticos e sistemas de suporte à decisão.

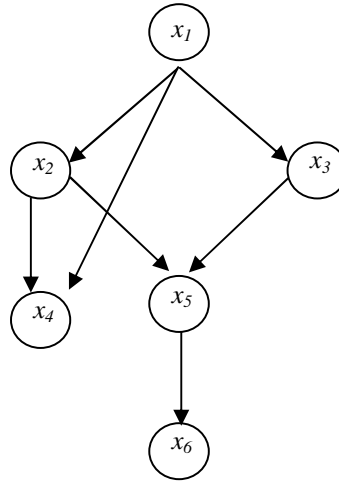
Uma rede *Bayesiana* consiste de dois componentes importantes:

- i) Um grafo direcionado acíclico (DAG)  $G = (V, E)$  – ou estrutura gráfica –, onde  $V = \{X_1, X_2, \dots, X_n\}$  é o conjunto de nós e representa as variáveis aleatórias e  $E$  é o conjunto de pares ordenados de elementos distintos de  $V$  e representa as relações de dependências entre as  $n$  variáveis. Os elementos de  $E$  são chamados de arestas (ou arcos). Uma aresta direcionada do nó  $X_i$  para o nó  $X_j$  indica que  $X_i$  é um dos pais de  $X_j$ .
- ii) Uma tabela de probabilidade condicional (CPT) – ou parâmetros numéricos –, que quantifica os efeitos que o conjunto de pais de  $X_i$  tem sobre as variáveis  $X_i$  em  $G$ .

Numa rede *Bayesiana*, a ausência de alguns arcos no DAG (conhecido como componente qualitativo) demonstra a existência de relações de independência condicional entre as variáveis e a presença deles pode representar a existência de relações de dependência direta. A tabela de probabilidades (conhecida como componente quantitativo) é uma coleção de medidas de probabilidades condicionais, as quais demonstram a força das dependências e são atualizadas com o uso do Teorema de Bayes [10], com base em uma nova informação amostral.

A Figura 2.1 apresenta um exemplo de rede *Bayesiana* e mostra a relação direta existente entre suas variáveis, ligadas por arcos orientados. Observa-se que  $x_i$  está diretamente relacionada com  $x_j$  se houver um arco de  $x_i$  para  $x_j$ . Alguns autores consideram a relação entre as variáveis como sendo uma relação causal [1] e esta relação tem as seguintes possibilidades:

- Uma variável pode causar uma ou mais variáveis (filhas).
- Uma variável pode sofrer a influência causal de uma ou mais variáveis (pais).



**Figura 2.1. Estrutura de Rede Bayesiana.**

A quantificação da relação causal é atualizada por uma distribuição de probabilidade condicional, a qual condiciona a variável causada à(s) sua(s) causadora(s) –  $P(\text{causada/causadora})$ . Representa-se a distribuição conjunta como o produto das distribuições condicionais dadas pelas relações causais na rede. Assim, para a rede da Figura 2.1, tem-se a fórmula (2.1):

$$\begin{aligned}
 P(x_1, x_2, \dots, x_6) &= P(x_6 / x_5) P(x_5 / x_2 x_3) P(x_4 / (x_1 x_2)) P(x_3 / x_1) P(x_2 / x_1) P(x_1) = \\
 &= \prod_{i=1}^m P(x_i / \pi_{x_i})
 \end{aligned}
 \tag{2.1}$$

na qual  $m$  é o número de variáveis,  $P(x_1, x_2, \dots, x_6)$  é a probabilidade conjunta de  $X = \{x_1, x_2, \dots, x_6\}$ , sendo  $X$  um conjunto de variáveis aleatórias,  $P(x_i / x_j)$  é a probabilidade condicional de  $x_i$ , dado que se conhece  $x_j$ , e  $\pi_{x_i}$  é o conjunto de pais de  $x_i$ .

Num modelo causal, não há relações com complexidade muito grande e a estrutura de cálculos terá uma complexidade proporcional ao número de variáveis dependentes e não ao número de variáveis do problema [9].

## 2.2 Inferência Bayesiana

Uma vez que a rede Bayesiana esteja definida, podem-se extrair conhecimentos representados nela através de um processo chamado inferência. A inferência Bayesiana calcula a distribuição de probabilidade condicional de um conjunto de variáveis de consulta, de acordo com os valores de um conjunto de variáveis de evidência.

Durante muitos anos, a probabilidade condicional de eventos de interesse tem sido calculada a partir de probabilidades conhecidas, usando o Teorema de *Bayes* (Teorema 2.1), apresentado a seguir (extraído de [11]).

*Teorema 2.1. (Bayes) Dado dois eventos E e F tal que  $P(E) \neq 0$  e  $P(F) \neq 0$ , tem-se:*

$$P(E | F) = \frac{P(F | E)P(E)}{P(F)} \quad (2.2)$$

sendo  $P(E|F)$  a probabilidade a posteriori (conhecimento a ser descoberto),  $P(F|E)$  a probabilidade condicional,  $P(E)$  a probabilidade a priori (conhecimento existente) de E,  $P(F)$  a probabilidade a priori de F.

O teorema de *Bayes* é usado quando não é possível determinar diretamente a probabilidade condicional de interesse, mas é possível determinar as probabilidades do lado direito da equação (2.2). O exemplo 2.1 (adaptado de [11]) ilustra como o teorema de *Bayes* é aplicado para calcular a probabilidade de um evento de interesse a partir de probabilidades conhecidas.

**Exemplo 2.1.** Suponha que João tenha feito um exame de raio-X do tórax, necessário para todos os novos empregados do banco onde ele trabalha, e que o resultado tenha sido positivo para câncer de pulmão. Então, João entra em pânico, certo de que ele tem câncer de pulmão. Mas ele deveria? Sem conhecer a precisão do teste, João realmente não tem como saber o quão provável é o fato de ele ter câncer de pulmão. Quando ele descobre que o teste não é absolutamente conclusivo, ele decide investigar sua precisão e ele aprende que há uma taxa de falso negativo de 0.4 e uma taxa de falso positivo de 0.02. Esta precisão é representada assim. Primeiro, definem-se as variáveis aleatórias:

Variável	Valor	Quando a variável tem esse valor
Teste	Positivo	Raio-X é positivo.
	Negativo	Raio-X é negativo.
Câncer de Pulmão	Presente	Câncer de Pulmão está presente.
	Ausente	Câncer de Pulmão está ausente.

Tem-se então estas probabilidades condicionais:

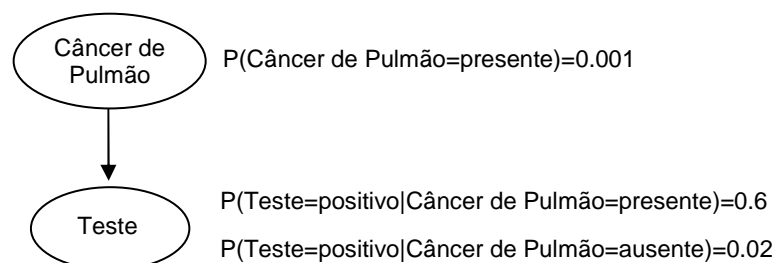
$$P(\text{Teste}=\text{positivo}|\text{Câncer de Pulmão}=\text{presente})=0.6$$

$$P(\text{Teste}=\text{positivo}|\text{Câncer de Pulmão}=\text{ausente})=0.02$$

Dado estas probabilidades, João sente-se um pouco melhor. Contudo, ele percebe que ainda não sabe o quão provável seja que ele esteja com câncer de pulmão. Ou seja, a probabilidade de João ter câncer de pulmão é  $P(\text{Câncer de Pulmão}=\text{presente}|\text{Teste}=\text{positivo})$ , e esta não é uma das probabilidades listadas acima. Finalmente, João lembra-se do teorema de *Bayes* e percebe que ainda precisa de outra probabilidade, a  $P(\text{Câncer de Pulmão}=\text{presente})$ , que é a probabilidade de se ter câncer de pulmão antes de qualquer informação sobre os resultados do teste. Mesmo esta probabilidade não sendo baseada em qualquer informação a respeito dos resultados do teste, ela é baseada em alguma informação. Especificamente, ela é baseada em toda informação (relevante para o câncer de pulmão) conhecida sobre João antes que ele pegasse o teste. A única informação sobre João, antes do teste, era que ele era um dos empregados que faria o teste, necessário para todos os novos empregados. Então, quando ele aprende que apenas 1 dos 1000 novos empregados tem câncer de pulmão, ele atribui 0.001 a  $P(\text{Câncer de Pulmão}=\text{presente})$ . Uma rede *Bayesiana* representando esta distribuição de probabilidade é mostrada na Figura 2.2. Assim, João aplica o teorema de *Bayes* como abaixo:

$$\begin{aligned}
 &P(\text{presente} | \text{positivo}) \\
 &= \frac{P(\text{positivo} | \text{presente})P(\text{presente})}{P(\text{positivo} | \text{presente})P(\text{presente}) + P(\text{positivo} | \text{ausente})P(\text{ausente})} \\
 &= \frac{(0.6)(0.001)}{(0.6)(0.001) + (0.02)(0.999)} \\
 &= 0.029
 \end{aligned}$$

Desta forma, João descobre que a probabilidade dele ter câncer de pulmão é apenas 0.03 e ele relaxa um pouco enquanto espera por resultados de testes futuros.



**Figura 2.2.** Rede *Bayesiana* representando a distribuição de probabilidade discutida no Exemplo 2.1.

O exemplo 2.2 ilustra como a probabilidade *a priori* pode mudar, dependendo da situação modelada.

**Exemplo 2.2.** Agora suponha que Samuel também esteja fazendo o exame de raio-X do tórax. Porém, ele está fazendo o exame porque trabalha nas minas há 20 anos e seus padrões ficaram preocupados quando souberam que aproximadamente 10% dos trabalhadores desenvolvem câncer de pulmão depois de muitos anos nas minas. O resultado do teste de Samuel também foi positivo. Qual é a probabilidade dele ter câncer de pulmão? Com base na informação conhecida sobre Samuel antes do teste, atribui-se uma probabilidade *a priori* de 0.1 a ele de ter câncer de pulmão. Novamente usando o teorema de Bayes, conclui-se que a  $P(\text{Câncer de Pulmão}=\text{presente}|\text{Teste}=\text{positivo})=0.769$  para Samuel. Assim, conclui-se que é muito provável que Samuel tenha câncer de pulmão.

A inferência *Bayesiana* é razoalmente simples quando envolve apenas duas variáveis relacionadas como nos exemplos 2.1 e 2.2. Contudo, torna-se muito mais complexo quando se deseja fazer inferência com muitas variáveis relacionadas. Considere a situação onde várias características estão relacionadas por meio de cadeias de inferência. Por exemplo, se ou não um indivíduo, que tem um histórico de fumante, tem uma influência direta sobre se ou não esse indivíduo tem bronquite e se ou não esse indivíduo tem câncer de pulmão. Por sua vez, a presença ou ausência de cada uma destas doenças tem uma influência direta sobre se ou não o indivíduo sente cansaço. Nesta situação, deseja-se fazer inferência probabilística envolvendo características que não estão relacionadas através de uma influência direta. Portanto, estas probabilidades condicionais não podem ser calculadas usando uma aplicação simples do teorema de Bayes.

Redes *Bayesianas* foram desenvolvidas para tratar estas dificuldades. De uma maneira geral, a inferência em redes *Bayesianas* permite responder consultas sobre um domínio de dados e tem como base um processo chamado propagação de evidências.

A inferência pode ser realizada apenas para se extrair um conhecimento existente na rede (*a priori*) ou, então, para se verificar o que ocorre quando determinadas situações acontecem. Quando o objetivo é a obtenção do conhecimento *a priori*, basta consultar a probabilidade *a priori* da variável alvo. Já quando o objetivo é a obtenção de valores em uma situação específica, deve-se fornecer à rede *Bayesiana* os fatos necessários (evidência), propagar estes fatos por toda a rede e, em seguida,

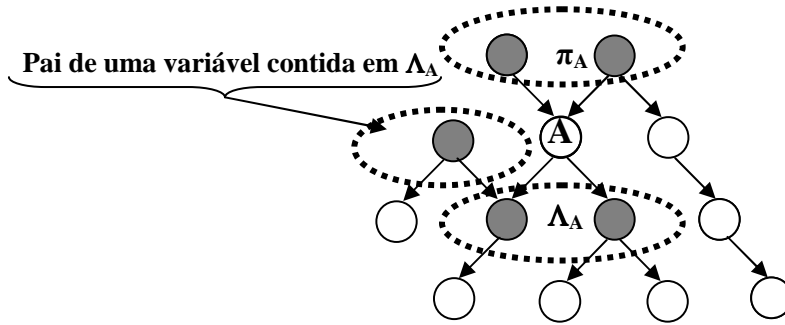




$X$ ,  $MB(X)$ , é qualquer conjunto de variáveis, tal que  $X$  seja condicionalmente independente de todas as outras variáveis dado  $MB(X)$ .

*Teorema 2.2. Suponha uma rede Bayesiana  $G$  que satisfaça a condição de Markov dada na Definição 2.1. Para cada variável  $X$  em  $G$ , o conjunto de todos os pais de  $X$ , filhos de  $X$  e pais dos filhos de  $X$ , é uma cobertura de Markov de  $X$  ( $MB(X)$ ).*

Resumindo a Definição 2.1 e o Teorema 2.2, considere uma rede *Bayesiana* onde  $\Lambda_A$  é o conjunto de filhos do nó (variável)  $A$  e  $\pi_A$  é o conjunto de pais da variável  $A$ . O conjunto de nós formado pela união dos conjuntos  $\Lambda_A$ ,  $\pi_A$  e os pais das variáveis contidas em  $\Lambda_A$  é chamado de Cobertura de *Markov* da variável  $A$ . Um exemplo genérico que ilustra a cobertura de *Markov* da variável  $A$  pode ser visto na Figura 2.4.



**Figura 2.4.** Os nós hachurados representam a Cobertura de *Markov* do nó  $A$ .

A prova do Teorema 2.2 é direta e é dada em [11], assim não será replicada aqui. A Definição 2.1 e o Teorema 2.2 motivam o Corolário 2.1.

*Corolário 2.1.* Dado uma rede *Bayesiana*  $G$ , os únicos nós em  $G$  que têm influência sobre a distribuição condicional de um dado nó  $X$  são os nós que formam a cobertura de *Markov* de  $X$  ( $MB(X)$ ).

O Corolário 2.1 é uma consequência simples da Definição 2.1 e Teorema 2.2 e também é provado em [11].

Como pode ser visto em [1], os únicos nós da rede que possuem influência sobre o cálculo da distribuição de probabilidade condicional do nó  $A$  (dados os estados de todos os outros nós da rede) são os contidos na Cobertura de *Markov* de  $A$ .

## 2.4 Aprendizado de Redes Bayesianas a partir de Dados

Redes Bayesianas podem ser modeladas a partir do conhecimento de especialistas do domínio ou a partir de dados. O aprendizado de redes Bayesianas a partir de dados surgiu como alternativa ao método de obter opiniões de especialistas, pois muitas dificuldades eram encontradas.

O processo de entrevistar o especialista para extrair conhecimento especializado é difícil e consome tempo. Além disso, nem sempre o engenheiro do conhecimento conhece a área de aplicação (domínio) em questão, o que dificulta muito a aquisição de informações. Portanto, o desenvolvimento de métodos automáticos capazes de aprender a rede diretamente a partir de dados tornou-se necessário.

O processo de aprendizado de redes Bayesianas divide-se em duas etapas: i) identificar a sua estrutura, ou seja, identificar as relações de interdependência dadas pelos arcos, e ii) induzir as distribuições de probabilidades da rede. Normalmente, este processo é dividido em dois subprocessos: “aprendizado da estrutura” e “aprendizado dos parâmetros numéricos” [13].

O aprendizado da estrutura é a identificação das dependências e independências das variáveis e da direção da causalidade, sendo considerado um problema difícil, principalmente devido ao fato de que o número de possíveis estruturas para um dado problema cresce exponencialmente de acordo com o número de variáveis. O aprendizado dos parâmetros numéricos, ou das probabilidades, é a identificação da “força” dos relacionamentos através da probabilidade condicional para cada nó, depois de fornecidos a estrutura e os dados. Aprender os parâmetros numéricos, contudo, pode ser considerado uma tarefa mais simples.

Diversos métodos têm sido desenvolvidos para tratar o problema do aprendizado de estrutura. De uma maneira geral, pode-se dizer que os métodos Bayesianos de aprendizado de estrutura dividem-se em duas classes principais. A primeira é a classe dos algoritmos que geram a rede através de uma busca heurística em uma base de dados, e alguns exemplos podem ser encontrados em [14], [15], [16], [17], [18], [19], [20] e [21]. Já na segunda classe, estão os algoritmos que utilizam o conceito de independência condicional [1] para a construção da rede; trabalhos que aplicam esta classe de algoritmos podem ser encontrados em [5], [22], [23], [24], [25] e [26]. Como não se pode definir uma classe como sendo a melhor, existem ainda trabalhos que

implementam versões híbridas que combinam as duas classes; alguns exemplos podem ser vistos em [21], [22], [23] e [24].

### 2.4.1 Métodos com base em Busca Heurística

Os algoritmos de busca heurística vêem o problema de aprendizado como um problema de busca pela estrutura que melhor represente os dados. Tradicionalmente, isto é feito aplicando um mecanismo de busca com algum critério de informação para medir a qualidade e a diferença entre as estruturas candidatas ao longo do espaço de busca.

Alguns métodos analisam uma única estrutura de rede por vez e a modificam iterativamente até uma boa solução, ou um critério de parada ser alcançado. Há também métodos que consideram um grupo de estruturas de redes por vez ao invés de uma única estrutura.

Para encontrar a melhor estrutura, há dois tipos de algoritmos de busca: *forward* (inicia com um grafo desconectado e vai adicionando, ou revertendo, os arcos) e *backward* (inicia com um grafo totalmente conectado e vai removendo, ou revertendo, os arcos). Para adicionar ou remover os arcos do grafo, os algoritmos usam algum tipo de busca, como busca heurística, para tornar a busca mais rápida (por exemplo, *Hill Climbing*, *Simulated Annealing*, etc).

As estruturas candidatas são avaliadas de acordo com um critério de pontuação. O processo é repetido até que a melhor estrutura seja encontrada. Há uma grande quantidade de trabalhos relacionados a funções de pontuação e algoritmos de busca usados para este propósito. Exemplos de métricas de pontuação incluem a métrica AIC, Bdeu, a métrica *Bayesiana*, a métrica CH, o *Minimum Description Length* (MDL), etc. Mais detalhes destas métricas podem ser encontrados em [16], [18], [27] e [28].

Um bom exemplo desta abordagem de aprendizado de redes *Bayesianas* é o algoritmo K2, proposto por *Cooper* e *Herskovitz* [16] e apresentado no tópico abaixo.

#### Algoritmo K2

K2 [16] é um algoritmo de busca heurística criado para o aprendizado de estruturas de redes *Bayesianas*. Ele recebe, como entrada, um conjunto de dados e uma

ordenação das variáveis e gera, como saída, a estrutura da rede. Este algoritmo é muito conhecido devido ao seu desempenho em termos de complexidade computacional (tempo) e resultados precisos, obtidos quando uma ordenação de variáveis adequada é fornecida [26][29].

A suposição dos atributos pré-ordenados é usada para reduzir o número de possíveis estruturas a serem aprendidas. K2 usa uma lista ordenada (contendo todos os atributos) que afirma que apenas os atributos posicionados antes de um dado atributo  $x_i$  podem ser pais de  $x_i$ . Assim, o primeiro atributo na lista não tem pais, ou seja, é um nó raiz na rede *Bayesiana*.

O algoritmo usa um método guloso para encontrar a melhor estrutura. Ele se inicia supondo que todos os nós não têm pais. Então, a partir do segundo atributo da lista ordenada (o primeiro é um nó raiz), os possíveis pais são testados e aqueles que maximizam a probabilidade da estrutura estar de acordo com a base de dados são adicionados à rede. Este processo é repetido para todos os atributos até que a melhor estrutura possível seja encontrada.

A métrica (pontuação) aplicada pelo K2 para testar cada conjunto de pais possíveis, para cada atributo, é dada por  $g(i, \pi_i)$ :

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \quad (2.3)$$

onde cada atributo  $x_i$  ( $i = 1, \dots, n$ ) tem  $r_i$  possíveis valores  $\{v_{i1}, v_{i2}, \dots, v_{iri}\}$ . Cada atributo  $x_i$  tem um conjunto de pais  $\pi_i$  e  $q_i$  é o número de instâncias de  $\pi_i$ .  $N_{ijk}$  é o número de objetos no conjunto de dados  $D$ , onde  $x_i$  tem o valor  $v_{ik}$  e  $\pi_i$  é instanciado como  $w_{ij}$ , o qual representa a  $j$ -ésima instância relativa a  $D$  de  $\pi_i$ . Finalmente,  $N_{ij} = \sum N_{ijk}$ .

O algoritmo K2 pode ser descrito em uma forma algorítmica, como mostrado na Figura 2.5. Depois que a melhor estrutura foi construída, as probabilidades condicionais da rede podem ser determinadas. Isto é feito usando uma estimativa *Bayesiana* da probabilidade da estrutura da rede. Estimativa *Bayesiana* é também adotada em outros métodos de aprendizado *Bayesiano*, mas há maneiras alternativas de calcular essa probabilidade como, por exemplo, por análise de variância.

**Algoritmo K2:**

{Entrada: Um conjunto de  $n$  nós, uma ordenação dos nós, limite superior  $u$  sobre o número de pais que um nó pode ter e uma base de dados  $D$ , contendo  $m$  casos.}

{Saída: Para cada nó, a identificação dos pais do nó.}

```

1. for  $i := 1$  to  $n$  do
2.    $\pi_i := \{\}$ ;
3.    $Pold := g(i, \pi_i)$ ; {Equação (2.3).}
4.    $OKToProceed := true$ 
5.   while  $OKToProceed$  and  $|\pi_i| < u$  do
6.     let  $z$  be the node in  $Pred(x_i) - \pi_i$  that maximizes  $g(i, \pi_i \cup \{z\})$ ;
7.      $Pnew := g(i, \pi_i \cup \{z\})$ ;
8.     if  $Pnew > Pold$  then
9.        $Pold := Pnew$ ;
10.       $\pi_i := \pi_i \cup \{z\}$ ;
11.    else  $OKToProceed := false$ ;
12.  end {while};
13.  write ('Node:',  $x_i$ , 'Parents of this node:',  $\pi_i$ )
14. end {for};
15. end {K2};

```

Figura 2.5. Pseudocódigo do algoritmo K2, adaptado de [16].

## 2.4.2 Métodos com base em Independência Condicional

Para a classe de algoritmos que utilizam o conceito de independência condicional, é assumido que a estrutura da rede representa perfeitamente as dependências e independências no domínio de aplicação, isto é, uma afirmação de independência é representada por uma estrutura se, e somente se, ela for uma independência válida para o domínio. A validade de uma independência pode ser verificada realizando-se testes estatísticos, como, por exemplo, o qui-quadrado ( $\chi^2$ ) [11], utilizando uma base de dados que representa o domínio.

Um bom exemplo de algoritmo de aprendizado baseado em independência condicional é o algoritmo PC, proposto por *Spirites et al* [5] e apresentado no tópico abaixo.

### Algoritmo PC

O algoritmo PC [5] é baseado em testes estatísticos de independência condicional. Ele procura por uma rede *Bayesiana* que represente as relações de independência entre as variáveis de um conjunto de dados. Isto é feito de acordo com o critério de independência condicional  $I(X_i; X_j/A)$ , definido em [1], onde  $A$  é um

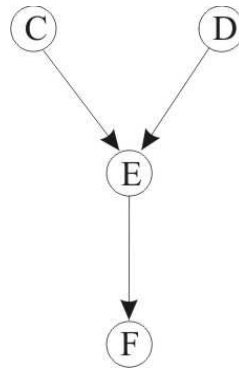
subconjunto de variáveis,  $X_i; X_j$  são variáveis. Se  $I(X_i; X_j/A)$  é verdadeiro, a variável  $X_i$  é condicionalmente independente de  $X_j$ , dado  $A$  (critério de d-separação).

Segundo [11], dado um conjunto de independências condicionais em uma distribuição de probabilidade, tenta-se encontrar um grafo direcionado acíclico (DAG), onde a condição de *Markov* envolve todas e apenas estas independências condicionais. A condição de *Markov*, definida em [11], estabelece o seguinte:

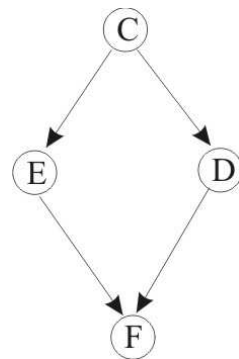
**Condição de Markov:** *Seja  $G = (V, E)$  um DAG, no qual  $V$  é um conjunto de vértices e  $E$  é um conjunto de arcos. Suponha que há uma distribuição de probabilidade conjunta  $P$  de variáveis aleatórias no conjunto  $V$ . Diz-se que  $(G, P)$  satisfaz a condição de Markov se para cada variável  $X \in V$ ,  $\{X\}$  é condicionalmente independente do conjunto de todos os seus não-descendentes, dado o conjunto de todos os seus pais.*

Suponha que se possa determinar (ou ao menos estimar) as independências condicionais,  $IND_P$ , em uma distribuição de probabilidade  $P$ . Geralmente, isto é feito a partir de dados. Como interessa apenas as independências condicionais entre conjuntos disjuntos, é suposto que  $IND_P$  contém todas e apenas estas independências condicionais. Em [11], define-se que a condição de *Markov* envolve todas e apenas estas independências condicionais que são identificadas por d-separação. Assim, o objetivo é encontrar um DAG cujas d-separações sejam as mesmas que em  $IND_P$ .

A d-separação é uma propriedade do DAG. Como definido em [11], seja  $G = (V, E)$  um DAG,  $A \subseteq V$ , e  $X$  e  $Y$  serem nós distintos em  $V-A$ . Diz-se que o nó  $X$  e nó  $Y$  são d-separados por um conjunto de nós  $A$  em  $G$  se toda ligação entre  $X$  e  $Y$  for bloqueada por  $A$ . Isto é, toda ligação de  $X$  para  $Y$  precisa necessariamente passar pelo conjunto de nós  $A$ . Na Figura 2.6, por exemplo, o conjunto de nós  $\{C, D\}$  é d-separado do nó  $F$  por um conjunto de nós formado pelo nó  $E$ . Neste caso, a d-separação é de ordem 1, pois o conjunto composto pelo nó  $E$  é formado por apenas um nó. Na Figura 2.7, a d-separação é de ordem 2, pois o nó  $C$  é d-separado do nó  $F$  por um conjunto formado por 2 nós,  $E$  e  $D$ .



**Figura 2.6.** GDA onde o conjunto de nós  $\{C, D\}$  é d-separado do nó  $F$  pelo nó  $E$ .



**Figura 2.7.** GDA onde o nó  $C$  é d-separado do nó  $F$  pelos nós  $\{E, D\}$ .

PC primeiro analisa os subconjuntos de ordem 0 (ou seja, os subconjuntos que possuem d-separação de tamanho 0), depois o subconjunto de ordem 1 (subconjuntos que possuem d-separação de tamanho 1), depois o subconjunto de ordem 2 (subconjuntos que possuem d-separação de tamanho 2) e assim sucessivamente. Considerando o DAG como a estrutura de uma rede *Bayesiana*, este processo pode ser utilizado para induzir uma estrutura de rede *Bayesiana*.

A Figura 2.8, adaptada de [5], apresenta o algoritmo PC. O conjunto de variáveis condicionadas deve pertencer ao conjunto de variáveis adjacentes. Assim, suponha que  $Adjacencies(B_s, A)$  seja o conjunto de vértices adjacentes à  $A$  no grafo direcionado  $B_s$ , que representa a estrutura de uma rede *Bayesiana*. Este algoritmo recebe, como entradas, uma base de dados e um conjunto de d-separações *IND* de ordens 0, 1, 2, e subseqüentes.

A hipótese de pré-ordenação poderia ser usada no PC nos passos de orientação dos arcos (passos C e D). Assim, uma lista ordenada (contendo todos os atributos) afirmaria que apenas os atributos posicionados antes de um determinado atributo  $X$  poderiam ser pais de  $X$ . Portanto, o uso de uma ordenação de variáveis pré-definidas



poderia ignorar a busca pela orientação dos arcos (como conduzido nos passos C e D, da Figura 2.8).

**Algorithm PC:**

```

A) Forma o grafo não direcionado completo  $B_s$  a partir do conjunto de vértices  $V$ .
B)
   $n = 0$ .
  repete
    repete
      seleciona um par ordenado de variáveis  $X$  e  $Y$  que são adjacentes
      em  $B_s$  tal que  $Adjacencies(B_s, X) \setminus \{Y\}$  tem cardinalidade maior ou
      igual a  $n$ , e um subconjunto  $S$  de  $Adjacencies(B_s, X) \setminus \{Y\}$  de
      cardinalidade  $n$ , e se  $X$  e  $Y$  são d-separados dado  $S$ , delete a
      aresta  $X - Y$  de  $B_s$  e grave  $S$  em  $Sepset(X, Y)$  e  $Sepset(Y, X)$ ;
    enquanto todos os pares ordenados de variáveis adjacentes  $X$  e  $Y$ , tal
    que  $n = n + 1$ ;
  enquanto para cada par ordenado de vértices  $X, Y$ ,  $Adjacencies(B_s, X) \setminus \{Y\}$  tem
  cardinalidade menor que  $n$ .
C) Para cada tripla de vértices  $X, Y, Z$  tal que o par  $X, Y$  e o par  $Y, Z$  são adjacentes
em  $B_s$ , mas o par  $X, Z$  não é adjacente em  $B_s$ , oriente  $X - Y - Z$  como  $X \rightarrow Y \leftarrow Z$ ,
se e somente se,  $Y$  não está em  $Sepset(X, Z)$ .
D) repete
  Se  $A \rightarrow B$ ,  $B$  e  $C$  são adjacentes,  $A$  e  $C$  não são adjacentes, e não há ponta
  de seta em  $B$ , então oriente  $B - C$  como  $B \rightarrow C$ .
  Se há um caminho direcionado de  $A$  para  $B$ , e uma aresta entre  $A$  e  $B$ ,
  então oriente  $A - B$  como  $A \rightarrow B$ .
enquanto não houver mais arestas a serem orientadas.

```

Figura 2.8. Pseudocódigo do algoritmo PC [5].

## 2.5 Aprendizado de Classificadores *Bayesianos* a partir de Dados

A indução de classificadores a partir de um conjunto de dados de instâncias pré-classificadas é um problema importante em aprendizado de máquina. Recentemente, o uso de redes *Bayesianas* para problemas de classificação tem recebido crescente atenção.

Diversas vantagens de redes *Bayesianas* podem torná-las atrativas para problemas de classificação. Por exemplo, a representação explícita de relações probabilísticas pode explorar a estrutura do problema, facilitando a incorporação de conhecimentos do domínio na concepção do modelo. Além disso, uma rede *Bayesiana* tem uma representação gráfica intuitiva que beneficia a decomposição de problemas grandes e complexos em problemas menores.

A aplicação de redes *Bayesianas* para classificação pode ser muito simples. Considere, por exemplo, que uma rede *Bayesiana* seja induzida a partir de dados,

usando o algoritmo K2 ou PC. Uma vez que a rede *Bayesiana* tenha sido induzida, pode-se utilizá-la com o fim específico de inferir o comportamento de uma única variável  $x_i$ . Neste caso, pode-se considerar  $x_i$  como classe. Quando este processo é executado, diz-se que se construiu um classificador *Bayesiano* irrestrito, ou seja, uma rede *Bayesiana* irrestrita está sendo utilizada para uma tarefa de classificação. No entanto, a indução de classificadores baseada em redes *Bayesianas*, por sua vez, pode gerar *overhead* computacional indesejáveis. Por isso, têm sido desenvolvidos algoritmos específicos para construção de classificadores *Bayesianos* que distinguem a variável classe desde o início do processo de indução a partir dos dados.

Quando uma rede *Bayesiana* é construída para propósitos de classificação, é possível impor restrições adicionais que geram ganhos extras de eficiência computacional. Tais restrições são normalmente especificadas com foco na precisão de classificação, sem levar em consideração outros aspectos importantes para um bom modelo de classificação. O *Naive Bayes* (NB) é um exemplo de um classificador induzido baseado em uma forte restrição. Ele pode ser visto como uma rede *Bayesiana* particular onde qualquer variável tem seus nós correspondentes conectados apenas ao nó representando a variável classe. Embora o NB tenha fornecido bons resultados em vários domínios [30], suas estimativas de probabilidades não são realísticas e seu desempenho de classificação pode ser melhorado. Além disso, o modelo NB induzido não pode capturar as relações reais entre as variáveis.

Sob esta perspectiva, alguns trabalhos buscam por classificadores mais sofisticados, como em [31], [32] e [33], aplicando uma alternativa promissora que envolve o relaxamento da restrição imposta na construção da estrutura do NB. O algoritmo TAN [30] é um bom exemplo do relaxamento dessa restrição (veja Subseção 2.5.1).

Uma abordagem diferente é adotada em [34] para induzir um classificador. O algoritmo desenvolvido, chamado de Markov-PC, é inspirado no PC (Subseção 2.4.2) e no conceito de Cobertura de *Markov* (Subseção 2.3), com o objetivo de induzir um classificador *Bayesiano* com um número reduzido de variáveis.

### 2.5.1 TAN

O algoritmo TAN (*Tree augmented naive Bayes*) [30] foi introduzido por *Friedman* e *Goldszmidt* com o objetivo de obter melhores classificadores que o

algoritmo *Naive Bayes* (NB) [35]. Para isto, o algoritmo TAN relaxa a restrição imposta na construção da estrutura do NB e permite representar dependências entre os pares de atributos.

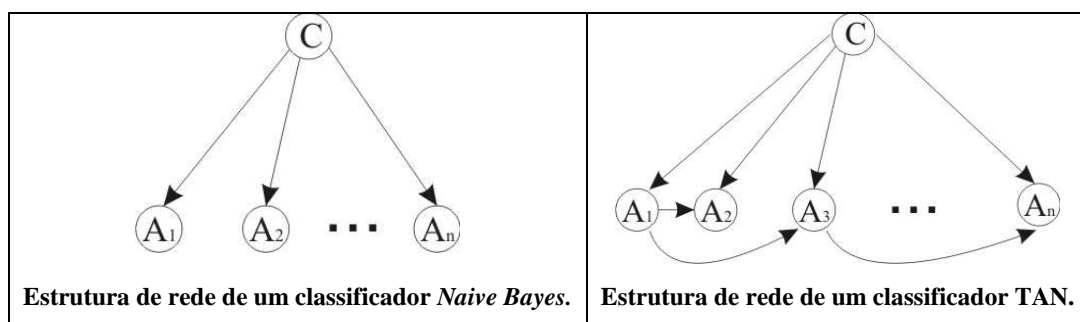
Os classificadores TAN consideram as dependências entre os outros atributos, além do atributo classe  $C$ . Estes modelos representam as relações entre os atributos  $A_1, \dots, A_n$ , condicionados ao atributo classe  $C$ , usando uma estrutura de árvore. As restrições impostas pelo algoritmo TAN são as seguintes:

- cada atributo depende condicionalmente da classe (da mesma forma como ocorre no *naive Bayes*);
- existem  $(n-1)$  atributos que dependem condicionalmente de outro atributo.

Esta última condição implica que, caso haja uma ligação de  $A_i$  para  $A_j$ , estes dois atributos não serão independentes, dada a classe. Em vez disso, a influência de  $A_j$  na probabilidade da classe  $C$  depende do valor de  $A_i$ .

A Figura 2.9 mostra duas estruturas de redes: uma para o NB e outra para o TAN. Nas duas estruturas, a classe encontra-se representada por  $C$  e os  $n$  atributos por  $A_j \forall 1 \leq j \leq n$ . Comparando as duas estruturas, verifica-se que foram acrescentadas à rede do TAN algumas dependências entre os atributos.

Para encontrar as dependências entre os atributos, *Friedman* usou o algoritmo reportado por *Chow* e *Liu* [36], utilizado para a construção de árvores de dependências que maximizam a informação mútua entre as variáveis.



**Figura 2.9. Comparação entre as estruturas dos modelos NB e TAN.**

No caso específico da construção da árvore de dependências para o TAN, visto que todos os atributos são dependentes da classe, é necessário conhecer a quantidade de informação que o atributo  $X$  fornece sobre o atributo  $Y$ , dada a classe, utilizando a fórmula (2.4) da informação mútua condicional abaixo:

$$I_p(X;Y|C) = \sum_{x,y,c} P(x,y,c) \frac{P(x,y|c)}{P(x|c)P(y|c)} \quad (2.4)$$

onde  $I_p(X;Y|C)$  é a informação que  $X$  fornece sobre  $Y$  (ou o inverso), dado  $C$ , sendo esta informação calculada para todos os pares de atributos. Após o cálculo das informações entre todos os pares de atributos, é possível construir um grafo completo.

A partir do grafo completo, o objetivo seguinte é obter a árvore de dependências que maximize a informação mútua condicional entre os atributos. Para achar esta árvore, há dois algoritmos conhecidos: o algoritmo de *Kruskal* [37] e o de *Prim's* [38].

A seguir, são apresentados os passos necessários para o aprendizado de um modelo TAN:

1. Obter a informação mútua condicional (equação (2.4)) entre cada par de atributos e construir um vetor com essa informação. A Figura 2.10 (a) ilustra este passo;
2. Desenhar um grafo completo não orientado, tendo como nós os atributos e como custo das ligações a informação mútua condicional entre os atributos. Veja a Figura 2.10 (b);
3. Calcular a árvore que maximiza a informação mútua condicional, sem criar ciclos, utilizando o algoritmo de *Kruskal* ou de *Prim's*, como exibido pela Figura 2.10 (c);
4. Transformar a árvore não orientada em orientada, escolhendo como raiz a informação mútua mais alta. Este passo é mostrado pela Figura 2.10 (d).
5. Adicionar a classe como “pai” de todos os atributos, como visualizado na Figura 2.10 (e).

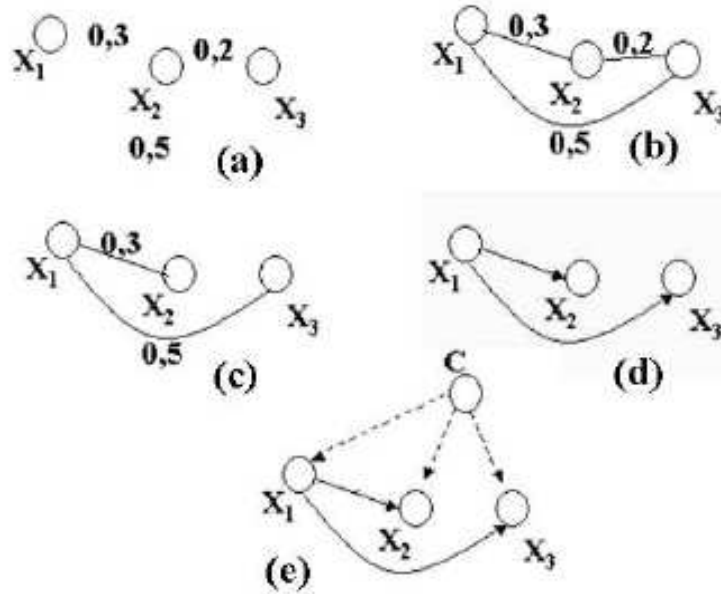


Figura 2.10. Fases do aprendizado de um modelo TAN.

## 2.6 O Problema do Aprendizado de Estruturas

O principal problema do aprendizado de estruturas está no aumento do espaço de busca, que cresce exponencialmente de acordo com o número de nós do domínio. Sendo assim, é considerado um problema NP-Completo [3][4] e não há métodos computacionais capazes de identificar a melhor solução para todos os problemas de aplicação. Desta forma, muitos algoritmos de aprendizado de estrutura são projetados para reduzir o espaço de busca.

A maioria das abordagens de busca e pontuação é executada no espaço de grafos direcionados acíclicos – DAGs – que representam as estruturas de redes *Bayesianas* possíveis [39]. O número de possíveis estruturas para um domínio com  $n$  variáveis é dado pela seguinte fórmula recursiva obtida por *Robinson* [40]:

$$f(n) = \sum_{i=1}^n (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i); f(0) = 1; f(1) = 1 \quad (2.5)$$

*Etminani et al.* [41] apresentam um algoritmo *branch and bound* que garante encontrar a estrutura de rede *Bayesiana* ótima global em menos tempo e requerimentos de memória, quando comparado aos melhores métodos exatos anteriores. O algoritmo usa uma função de pontuação que pode ser decomposta e drasticamente reduz o espaço

de busca de estruturas possíveis. Em [42], é possível encontrar outros trabalhos que exploram o espaço de DAGs.

Outras possibilidades incluem procurar no espaço de classes equivalentes de redes *Bayesianas* [43], quando a pontuação verifica a propriedade de equivalência de independências condicionais [44]. Métricas de pontuações como a BDe [18], que fornece o mesmo valor aos DAGs dentro da mesma classe, deveriam ser considerados.

Além do espaço de DAGs e do espaço de classes equivalentes de DAGs, outros espaços de busca podem ser usados. Um espaço que tem recebido alguma atenção é o espaço de ordenações das variáveis. Diversos trabalhos sobre aprendizado de redes *Bayesianas* têm utilizado a vantagem de se trabalhar com a ordenação das variáveis em vez de estruturas. Muitos deles encontrados em [42]. Embora o espaço de ordenações seja menor que o espaço de estruturas ( $n!$  em vez de  $2^{O(n^2)}$ ), não existem métodos automáticos computacionalmente eficientes na busca de uma ordenação de variáveis adequada para o aprendizado de redes *Bayesianas*.

### 2.6.1 A Ordenação de Variáveis

Na tentativa de reduzir o espaço de busca do processo de aprendizado de redes *Bayesianas*, algumas restrições podem ser impostas e, assim, os algoritmos podem gerar bons resultados com um esforço computacional aceitável. Uma restrição muito comum nos algoritmos de aprendizado de redes *Bayesianas* é a ordenação prévia das variáveis utilizadas na definição do problema [5].

Alguns algoritmos de aprendizado, baseados em busca heurística, tendem a utilizar a ordenação prévia das variáveis no processo de aprendizagem e sofrem grande influência desta ordenação, como é o caso do algoritmo K2 (Subseção 2.4.1). Esta dependência ocorre da seguinte forma. Considere uma base de dados com três atributos  $A_1$ ,  $A_2$  e  $A_3$ . Se o processo de aprendizado for realizado com os atributos ordenados desta maneira, isto implica que  $A_1$  não terá pais na rede, pois é o primeiro atributo, mas poderá ser pai de  $A_2$  e  $A_3$ ; já  $A_2$  poderá ter como pai  $A_1$  e poderá ser pai de  $A_3$ ; e finalmente  $A_3$  não terá filhos, mas poderá ter como pais  $A_1$  e  $A_2$ . Desta forma, pode-se definir a influência da ordem das variáveis no processo de aprendizado como se segue:

1. toda variável só pode ter como pais (na estrutura da rede) as variáveis que estão à sua esquerda na lista que define a ordenação das variáveis; e

2. toda variável só pode ter como filhos (na estrutura da rede) as variáveis que estão à sua direita na lista que define a ordenação das variáveis.

Os algoritmos de aprendizado, baseados no conceito de independência condicional, buscam definir a direção e o sentido dos arcos através da identificação das variáveis que são condicionalmente independentes do conjunto de variáveis do problema. Sendo assim, não exigem uma ordenação prévia das variáveis. Mas em alguns casos, podem existir arcos que não podem ter seu sentido definido, e nestes casos é necessário algum artifício extra para completar a definição da rede, como por exemplo, o auxílio de um especialista humano. Embora não exijam uma ordenação prévia, estes algoritmos são mais rápidos quando podem contar com tal ordenação, e, como foi mostrado em [45], possuem uma tendência de melhores resultados.

Nota-se então que, mesmo em algoritmos que não exigem uma ordenação prévia das variáveis, esta ordem pode auxiliar na otimização do processo de aprendizagem. É importante ressaltar que, da mesma forma que uma boa ordenação das variáveis pode melhorar os resultados do aprendizado de uma rede *Bayesiana*, uma ordem que não represente de forma correta o relacionamento das variáveis pode inserir erros no processo de aprendizagem.

A informação da ordenação de variáveis pode não estar disponível em aplicações do mundo real [46] e, muitas vezes, é apresentada de forma aleatória aos algoritmos de aprendizado, o que leva a resultados pobres [2].

A definição da ordenação de variáveis pode ser feita por um especialista humano, mas nem sempre há um especialista disponível e, realizar uma busca exaustiva através de todas as ordenações para problemas grandes, pode ser intratável. Assim, algumas heurísticas têm sido propostas para encontrar uma ordenação adequada.

Os primeiros trabalhos sobre este assunto foram apresentados por *Larrañaga et al.* [47], usando algoritmos genéticos para procurar através do espaço de ordenações, e *Wallace and Korb* [48] que usam um procedimento de amostragem MCMC (*Markov-chain Monte Carlo*). *Acid et al.* [49] usa uma abordagem similar a *Sing and Valtorta* [50], usando testes de independência condicional (CI). Contudo, em vez de aprender uma ordenação definitiva, uma busca é executada para preservar tantas CIs quanto possíveis. Outros trabalhos que usaram testes de CI para encontrar uma ordenação são *de Campos et al.* [51], *de Campos and Huete* [52] e *Chen et al.* [46].

## 2.7 Considerações Finais

Como uma consequência da melhora do poder computacional nas últimas décadas, estratégias de busca envolvendo algoritmos evolucionários [53] têm sido propostas para resolver o problema do aprendizado de estruturas de redes *Bayesianas*. Visto que técnicas de otimização clássica apenas exploram uma porção limitada do espaço de solução, pesquisadores logo perceberam que mecanismos de busca sequencial, que tentam melhorar uma única solução, eram claramente insuficientes para mover-se através destes grandes espaços de busca. O uso de mecanismos de busca aleatórios, com base em populações, foi proposto como uma alternativa que superaria essas limitações e seriam mais capazes de explorar o vasto espaço de soluções. Em [39] e [42], são apresentados alguns trabalhos que aplicam técnicas evolucionárias ao aprendizado de estruturas.

Partindo-se deste conceito, este trabalho foca na busca de uma ordenação de variáveis adequada para o aprendizado de redes *Bayesianas*, utilizando para isto algoritmos evolucionários. O Capítulo 3 aborda os conceitos iniciais dos algoritmos evolucionários e o Capítulo 4 apresenta o problema da ordenação de variáveis, destacando a aplicação daqueles algoritmos na solução deste problema.



## 3 Algoritmos Evolucionários

### 3.1 Definições

A computação evolucionária (ou evolutiva) é uma abordagem que se inspira em mecanismos evolucionários naturais para desenvolver algoritmos computacionais. Existem vários modelos computacionais evolucionários que foram propostos e estudados, e aos quais se dá a denominação de Algoritmos Evolucionários ou Evolutivos (AEs) [53]. Estes algoritmos têm em comum o fato de simularem a evolução de um conjunto de estruturas individuais, através de processos de seleção e reprodução que dependem do desempenho dessas estruturas num determinado ambiente.

Os AEs foram desenvolvidos com o objetivo de serem aplicados a problemas de otimização e variam conforme os vários modelos existentes de computação evolutiva [53]. As principais diferenças entre estes modelos estão na representação dos indivíduos e, conseqüentemente, no tipo de operadores genéticos utilizados. Existem ainda outras diferenças, como a ordem em que são executadas algumas das operações bem como os métodos de seleção utilizados.

Embora exista uma grande variedade de AEs, podem ser identificados aspectos que são comuns a todos eles. Em geral, apresentam os três componentes básicos seguintes, quando aplicados à solução de problemas de otimização:

- i. Conjunto de indivíduos que representam soluções candidatas ao problema, mantidas numa população;
- ii. Mecanismo de seleção, enfatizando a sobrevivência dos indivíduos que representam soluções de maior qualidade para o problema;
- iii. Mecanismo de exploração do espaço de busca (e controle da diversidade), conhecidos como operadores genéticos.

A Figura 3.1 apresenta um algoritmo típico (adaptado de [54]) que descreve os principais passos de qualquer AE. Inicialmente, é gerada uma população formada por um conjunto de indivíduos que podem ser vistos como possíveis soluções para o problema. Durante o processo evolutivo, a população é avaliada: para cada indivíduo, é dada uma nota, ou índice de aptidão, refletindo sua habilidade de adaptação a um determinado ambiente. Uma porcentagem dos mais adaptados é mantida, enquanto os outros são descartados. Os membros mantidos pela seleção podem sofrer modificações

em suas características fundamentais por meio dos operadores genéticos, gerando descendentes para a próxima geração. Os indivíduos com maior adaptação relativa têm maiores chances de se reproduzir. Para prevenir que os melhores indivíduos não desapareçam da população pela manipulação dos operadores genéticos, pode-se definir uma política elitista que os coloque na próxima geração. As iterações são repetidas até que seja encontrada uma solução satisfatória, de acordo com o critério de parada.

```

Algoritmo 1: Algoritmo Evolucionário
01:  $t = 0$ ;
02: Gerar População Inicial  $P(t)$ ;
03: Avaliar ( $P(t)$ );
04: enquanto Critério de parada não for satisfeito faça
05:    $P'(t) = \text{selecionar } (P(t))$ ;
06:    $P''(t) = \text{aplicar\_operadores\_genéticos } (P'(t))$ ;
07:    $P(t+1) = \text{criar\_população\_seguinte } (P(t), P''(t))$ ;
08:   avaliar ( $P(t + 1)$ );
09:    $t = t + 1$ ;
10: fim enquanto
11: Devolver melhor indivíduo;

```

**Figura 3.1.** Pseudocódigo de um típico algoritmo evolucionário (adaptado de [54]).

Uma vez descrito o algoritmo típico, é fácil identificar os aspectos que devem ser considerados quando se pretende utilizar um AE para resolver um determinado problema:

- i. Determinação do tipo de representação e codificação das soluções.
- ii. Definição da função de avaliação.
- iii. Escolha do método de seleção a ser aplicado.
- iv. Escolha dos operadores genéticos a serem aplicados.
- v. Definição dos parâmetros de controle do algoritmo como, por exemplo, o tamanho da população, o número máximo de gerações, ou a probabilidade de aplicação de cada um dos operadores genéticos.
- vi. Escolha de um critério de parada.

Um dos modelos de AEs mais conhecidos são os Algoritmos Genéticos (AGs), desenvolvidos por *John Holland* [55]. Estes algoritmos partem do pressuposto que indivíduos com boas características genéticas têm maiores chances de sobreviver e produzir indivíduos mais aptos em uma dada população.

Os AGs são capazes de identificar e explorar aspectos do ambiente onde o problema está inserido e convergir globalmente para soluções ótimas, ou

aproximadamente ótimas. Por isso, têm se mostrado muito eficientes como ferramentas de busca e otimização para a solução dos mais diferentes tipos de problemas [56].

A Figura 3.2 apresenta o diagrama geral de um AG e mostra que estes algoritmos são mais específicos que os AEs (Figura 3.1). Note que, nas linhas 6 e 7 da Figura 3.2, são aplicados operadores genéticos de cruzamento e mutação para compor a nova população.

Neste trabalho, os AEs e os AGs são empregados no problema da busca de uma ordenação adequada de variáveis adequada ao processo de aprendizado de redes *Bayesianas*. Nas próximas seções, serão descritos, com mais detalhes, a representação do problema, os métodos de seleção e os operadores clássicos de cruzamento e mutação.

```

Algoritmo 2: Algoritmo Genético
01:  $t = 0$ ;
02: Gerar População Inicial  $P(t)$ ;
03: Avaliar ( $P(t)$ );
04: enquanto Critério de parada não for satisfeito faça
05:    $P'(t) = \text{selecionar}(P(t))$ ;
06:    $P''(t) = \text{aplicar\_operador\_cruzamento}(P'(t))$ ;
07:    $P'''(t) = \text{aplicar\_operador\_mutação}(P''(t))$ ;
08:    $P(t+1) = \text{criar\_população\_seguinte}(P(t), P''(t), P'''(t))$ ;
09:   avaliar ( $P(t+1)$ );
10:    $t = t + 1$ ;
11: fim enquanto
12: Devolver melhor indivíduo;
  
```

Figura 3.2. Pseudocódigo do algoritmo genético.

## 3.2 Representação do Problema

O primeiro aspecto a ser considerado antes da utilização de algoritmos genéticos para a solução de um problema de busca ou otimização é a representação desse problema.

Os AEs processam populações de indivíduos (ou cromossomos). Cada um deles representa uma possível solução do problema a ser otimizado. Estes cromossomos são estruturas de dados, geralmente vetores ou cadeias de valores binários. Se o cromossomo representa  $n$  variáveis de uma função, então o espaço de busca é um espaço com  $n$  dimensões. A maioria das representações é genotípica, utilizam vetores de tamanho finito em um alfabeto também finito.

Tradicionalmente, o genótipo de um indivíduo é representado por um vetor binário, no qual cada elemento do vetor, chamado de *gene*, denota a presença (1) ou

ausência (0) de uma determinada característica. Os elementos podem ser combinados formando as características reais do indivíduo, ou seja, o seu fenótipo.

A representação dos cromossomos também pode ser por meio de números reais ou inteiros, sendo mais naturalmente compreendida pelo ser humano e requerendo menos memória que aquela usando uma cadeia de *bits*.

A utilização de representações em níveis de abstração mais altos tem sido investigada e por serem mais fenóticas, facilitariam seu uso em determinados ambientes. Neste caso, precisam ser criados os operadores específicos para utilizar essas representações [57].

### 3.3 Métodos de Seleção

Através de um critério de seleção, os AEs buscam gerar indivíduos mais aptos, depois de um determinado número de gerações, a partir de um conjunto de cromossomos iniciais.

Os AEs começam com uma população inicial de  $N$  cromossomos. A cada cromossomo da população é atribuído um valor dado por uma função  $f_{apt}$  denominada função de aptidão. Esta função recebe os valores dos genes do cromossomo como entrada e fornece sua aptidão como resultado final.

Uma função de aptidão é geralmente uma expressão matemática que mede o quanto uma solução está próxima ou distante da solução desejada. Alguns problemas de otimização procuram maximizar o valor da função de aptidão, outros procuram minimizar o seu valor.

Associada uma nota ou aptidão a cada indivíduo da população, o processo de seleção escolhe então um subconjunto de indivíduos da população atual, gerando uma população intermediária. Os métodos de seleção mais utilizados são: método da roleta, método do torneio e método da amostragem universal estocástica. Mais detalhes sobre estes métodos de seleção podem ser encontrados em [57].

### 3.4 Operadores Genéticos Clássicos

Os algoritmos genéticos utilizam um conjunto de operadores para gerar sucessivas populações a partir da população inicial. Estes operadores são: cruzamento (*crossover*) e mutação.

O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população. Ele altera arbitrariamente um ou mais componentes de uma estrutura escolhida, fornecendo meios para colocar novos elementos na população. Esse operador é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação ( $P_m$ ), que geralmente é uma taxa pequena, por exemplo ( $0,01 \leq P_m \leq 0,1$ ) [57], pois é um operador secundário.

O cruzamento é o operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso é aplicado com probabilidade dada pela taxa de cruzamento ( $0,6 \leq P_c \leq 0,99$ ) [57], que deve ser maior que a taxa de mutação.

### 3.5 Operadores Genéticos de Permutação

Existem problemas que dependem da ordem em que as ações são executadas. Tais problemas têm sido estudados na literatura de algoritmos genéticos e têm levado à proposta de vários operadores genéticos para realizar permutações [58]. A ordenação das variáveis para o aprendizado de redes *Bayesianas* é um bom exemplo de permutação.

Os operadores de mutação para permutações são relativamente simples. Na mutação baseada na posição, dois elementos do cromossomo são escolhidos aleatoriamente e o segundo elemento é colocado antes do primeiro. Na mutação baseada em ordem, dois elementos do cromossomo são escolhidos aleatoriamente e suas posições são trocadas. A mutação por embaralhamento começa escolhendo aleatoriamente dois cortes no cromossomo. Depois os elementos na sub-lista entre os cortes são embaralhados.

Existem vários operadores de cruzamento para permutação. Alguns deles, encontrados em [56] e [59], são: OBX (*Order-Based Crossover*), PBX (*Position-Based*

*Crossover*), PMX (*Partially Matched Crossover*), CX (*Cycle Crossover*) e OX (*Order Crossover*).

### 3.6 Parâmetros Genéticos

O desempenho de um AE é fortemente influenciado pela definição dos parâmetros a serem utilizados. Portanto, é importante analisar como os parâmetros podem ser utilizados diante das necessidades do problema e dos recursos disponíveis [57]. A seguir são discutidos alguns critérios para a escolha dos principais parâmetros:

- a. **Tamanho da População:** O tamanho da população afeta o desempenho global e a eficiência dos AEs. Em uma população pequena, o AE é mais rápido, porém tem pouca cobertura do espaço de soluções. Já numa população grande, o AG possui uma cobertura representativa do espaço de soluções, mas fica mais lento.
- b. **Taxa de Cruzamento:** Se a taxa de cruzamento for muito alta, novos indivíduos são introduzidos mais rapidamente na população, mas pode ocorrer perda de indivíduos mais aptos. Caso seja muito pequena, o AE pode tornar-se lento e a busca pode estagnar.
- c. **Taxa de Mutação:** Se a taxa for muito alta, pode tornar o algoritmo desprovido de direção na busca, tornando-a essencialmente aleatória. Uma baixa taxa pode fazer com que a busca fique estagnada em sub-regiões do espaço de busca.
- d. **Critério de Parada:** Diferentes critérios podem ser utilizados para terminar a execução do AE, por exemplo, parar após certo número de gerações consecutivas em que não se obtém aperfeiçoamento da solução.

### 3.7 MOS: *Multiple Offspring Sampling*

*Multiple Offspring Sampling* (MOS) é introduzido como uma variante de um algoritmo evolucionário tradicional com base em população. Em [8], MOS é apresentado como um algoritmo adaptativo híbrido que manipula simultaneamente

várias abordagens evolucionárias e dinamicamente ajusta a participação de cada uma delas no processo de busca.

Em MOS, o algoritmo principal manipula os mecanismos para produzir novos indivíduos de diferentes abordagens evolucionárias (operadores de recombinação em Algoritmos Genéticos, modelos probabilísticos em Estimativas de Algoritmos de Distribuição, Estratégias de Evolução ou Evolução Diferencial). MOS usa os mecanismos oferecidos por estas heurísticas para criar os descendentes para a próxima geração. Cada um destes mecanismos capazes de produzir uma nova descendência é chamado de técnica. Especificamente, uma técnica pode ser definido como [7]:

- um modelo evolucionário particular;
- com uma codificação apropriada;
- usando operadores específicos (se necessário), e
- configurado com seus parâmetros necessários.

Cada uma destas técnicas é capaz de produzir um subconjunto dos descendentes a partir da população atual, a qual é compartilhada por todas as técnicas. Neste contexto, a tupla  $(n, T, P, O)$  é chamada um sistema MOS, onde  $n$  é o número de técnicas no conjunto de técnicas  $T=\{T_i\}$ .  $P=\{P_i\}$  é o conjunto de tamanho  $m$  de populações comuns por geração e  $O=\{O_i^{(j)}\}$  é o conjunto  $n \times m$  de população descendente por técnica e geração. A Figura 3.3 apresenta um pseudocódigo de MOS, descrevendo o funcionamento geral desta abordagem híbrida.

A abordagem MOS propõe a definição de mecanismos múltiplos para gerar novos indivíduos e os fazem competir durante o processo de evolução. Cada mecanismo cria seus próprios descendentes  $O_i^{(j)}$  ( $i$  é a geração e  $j$  é a técnica). O cálculo da quantidade de novos indivíduos criados em cada geração ( $II_i^{(j)}$ ), para  $n$  diferentes métodos de amostragem de descendentes, é obtido usando uma “Função de Participação”. Estas funções podem realizar atribuições estáticas simples ou, mais interessante, ajustes dinâmicos de acordo com uma “Medida de Qualidade” ( $Q_i^{(j)}$ ), que avalia o quão bons são os descendentes de cada técnica, a partir do ponto de vista desta medida [7].

Neste sentido, diferentes medidas podem ser propostas, dependendo do conceito de qualidade a ser considerado. Uma opção seria considerar a média de aptidão do subconjunto dos melhores indivíduos da descendência gerada por cada técnica como

medida de qualidade. Outras alternativas poderiam ser usadas (por exemplo: idade, diversidade, etc).

A qualidade de cada uma das técnicas disponíveis é recalculada a cada geração, como descrito no passo 3 do algoritmo MOS (Figura 3.3).

Adicionalmente, várias funções de participação podem ser propostas. Estas funções calculam um fator de equilíbrio para cada técnica, o qual representa um decréscimo da participação para  $j$ -ésima técnica na  $i$ -ésima geração, para cada técnica, exceto aquela que executa melhor.

```

Algoritmo: Multiple Offspring Sampling
1  begin
2    Participação distribuída uniformemente entre as
     $n$  técnicas usadas.
3    Crie a população global de soluções candidatas
     $P_0$ . Cada técnica produz um subconjunto de
    indivíduos de acordo com sua participação
     $(\Pi_0^{(j)})$ .
4    Avalie a população inicial  $P_0$ .
5    while critério de término não é alcançado do
6      for para cada técnica disponível  $T_j$  do
7        while taxa  $\Pi_0^{(j)}$  não é excedida to
8          Crie novos indivíduos a partir da
          população  $P_i$ .
9          Avalie os novos indivíduos.
10         Adicione os novos indivíduos a uma
          população auxiliar  $O_i^{(j)}$ .
11        end
12        Atualize a qualidade de  $T_j \rightarrow Q_j^{(i)} = Q(O_i^{(j)})$ .
13      end
14      Combine populações  $O_i^{(j)}$  e  $P_i$  de acordo com
      um critério pré-estabelecido para gerar  $P_{i+1}$ .
15      Atualize as taxas de participação a partir dos
      valores de qualidade calculados no passo 3.
16    end

```

Figura 3.3. Pseudocódigo do algoritmo MOS [7].



## 4 O Problema da Ordenação de Variáveis

### 4.1 Considerações Iniciais

A informação sobre uma ordenação de variáveis adequada pode reduzir o espaço de busca significativamente numa tarefa de aprendizado. O resultado dos algoritmos de aprendizado, principalmente com base em busca heurística, pode sofrer grande influência desta ordenação. No entanto, encontrar tal ordenação para as variáveis em uma rede *Bayesiana* é um problema complexo, que exige muita informação sobre o modelo [52].

Os nós em uma rede *Bayesiana* admitem, pelo menos, uma ordenação com base na estrutura do DAG relacionado à rede. Esta ordenação topológica impõe a propriedade de que um nó  $X_i$  ser pai de  $X_j$  (no grafo) implica em  $X_i < X_j$  na ordenação. Em outras palavras, um nó pode apenas ser um pai de outro nó se o precede na ordenação dos nós.

A estrutura de uma rede *Bayesiana* depende bastante da ordenação de suas variáveis: dada qualquer ordenação, é sempre possível construir uma rede *Bayesiana* (isto é, um mapa de independência mínimo do modelo de dependência correspondente), onde os arcos são consistentes com a ordenação inicial; contudo, a topologia da rede e assim o número de relações de independência condicional, que pode ser explicitamente representado, pode variar muito de uma ordenação para outra. Como uma representação esparsa é sempre preferível à uma representação densa do mesmo modelo, a tarefa de determinar a ordenação que dará origem à rede, com um número mínimo de arcos, é importante.

A ordenação das variáveis influencia principalmente os algoritmos de aprendizado baseado em busca heurística. Alguns destes algoritmos dependem da ordenação das variáveis para determinar a direção dos arcos. Os métodos baseados em independência condicional tentam encontrar a direção dos arcos sem necessitar desta ordenação, mas quando é conhecida, os resultados dos algoritmos podem ser aperfeiçoados [2].

A definição da ordenação de variáveis pode ser feita por um especialista humano. Mas nem sempre há um especialista próximo quando se executa tarefas de mineração de dados e aprendizado de máquina. Desta forma, algoritmos de aprendizado

de redes *Bayesianas* a partir de dados são aplicados frequentemente usando uma ordenação aleatória das variáveis, ou uma ordenação dada pelo conjunto de dados, que podem levar a resultados pobres [2].

Encontrar uma ordenação adequada é um problema complexo. É importante dizer que uma busca exaustiva através de todas as ordenações para problemas grandes continua intratável ( $n!$  para um problema com  $n$  variáveis). Por exemplo, observe os domínios de dados apresentados na Tabela 4.1. Esta tabela exhibe o número de ordenações de variáveis possíveis (OVP) – considerado  $n!$  – o número de ordenações de variáveis ótimas (OVO) que permitirá a indução de tais estruturas e o número de ordenações de variáveis não ótimas (OVNO). Para estes domínios, as estruturas de RBs são conhecidas e, portanto, é possível calcular o número de OVO. Além disso, os domínios *Synthetic 50-50*, *Synthetic 50-100* e *Synthetic 50-200* foram criados com o mesmo número de variáveis (50), mas com número de arestas diferentes. Esta variação no número de arestas foi introduzida com a intenção de analisar o impacto destes números na identificação das OVO. Percebe-se que o número de OVO é bem inferior ao número de OVP e diminui ainda mais com o aumento no número de arcos.

**Tabela 4.1.** Descrição dos conjuntos de dados, constando o nome do domínio (DOMÍNIO), o número de atributos (AT), o número de ordenações possíveis (OVP), o número aproximado (por conta do arredondamento) de ordenações ótimas (OVO) e o número aproximado (por conta do arredondamento) de ordenações não ótimas (OVNO).

DOMÍNIO	AT	OVP	OVO	OVNO
Alarm	37	1.37e+43	1.66e+15	1.36e+43
Ásia	8	40320	42	40278
Engine	9	362880	12720	350160
Synthetic 1	32	2.63e+35	7.61e+26	2.62e+35
Synthetic 2	32	2.63e+35	5.03e+23	2.62e+35
Synthetic 3	32	2.63e+35	1.02e+25	2.62e+35
Synthetic 50-50	50	3.04e+64	8.31e+36	3.03e+64
Synthetic 50-100	50	3.04e+64	2.24e+25	3.04e+64
Synthetic 50-200	50	3.04e+64	3.93e+19	3.04e+64

## 4.2 Influência da Ordenação de Variáveis

A ordenação das variáveis não é fundamental no aprendizado de redes *Bayesianas*, mas é uma restrição que pode ser imposta para aprender estas redes de forma mais rápida. Neste trabalho, esta restrição é investigada, analisando onde ela predomina e como obtê-la da melhor maneira possível.

Existem vários critérios que definem se a rede *Bayesiana* foi bem aprendida ou não, como distância da rede original (quando ela existe) para a rede gerada (número de arcos ausentes, número de arcos adicionais, número de arcos com direção invertida), taxa de classificação, precisão nas estimativas de probabilidades, entre outros. Cada critério deve ser escolhido de acordo com o problema abordado e nem todos são capazes de refletir a influência da ordenação de variáveis. Por isso, é importante saber se a busca pela ordenação de variáveis é adequada para cada problema a ser investigado.

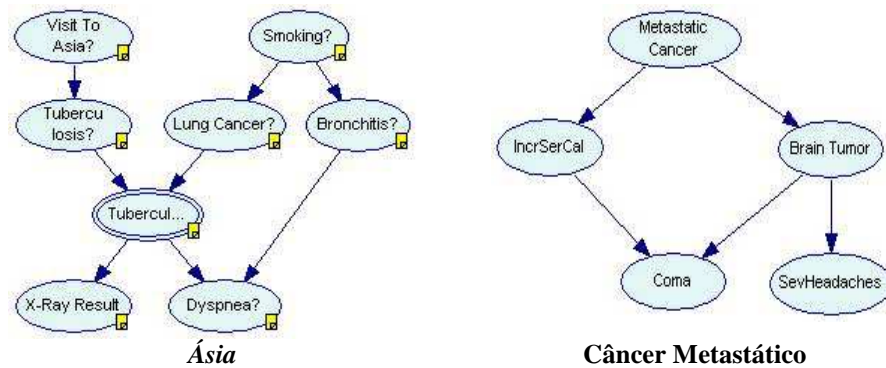
A influência da ordenação no aprendizado de redes *Bayesianas* pode ser observada segundo algum critério de avaliação das estruturas aprendidas. Nos itens abaixo, são apresentados três critérios que permitem esta observação.

#### 4.2.1 Valor da Função $g$

Uma possível forma de avaliar a estrutura de uma rede *Bayesiana* é utilizar o valor da função  $g$  (função (2.3)), calculado pelo algoritmo K2, que sofre grande influência da ordenação de variáveis. Como já explicado na Subseção 2.4.1, esta função possibilita maximizar a probabilidade da estrutura estar de acordo com a base de dados. Ao final do processo de aprendizado, a melhor estrutura aprendida representa o relacionamento direto entre as variáveis do domínio. Quanto maior for o valor de  $g$ , melhor será a estrutura aprendida.

Veja o exemplo de duas bases de dados conhecidas: *Asia* e Câncer Metastático, extraídas de [60]. Suas estruturas de redes Bayesianas originais são apresentadas na Figura 4.1. Suponha que estas bases de dados sejam fornecidas como entrada ao algoritmo K2, para três execuções, com três ordenações diferentes (mantendo uma variável como classe na primeira posição):

- i) uma ordenação ótima, na qual as possíveis variáveis pais vêm antes das possíveis variáveis filhas;
- ii) uma ordenação intermediária que não está tão distante da ordenação ótima;
- iii) e uma ordenação ruim, que está mais distante da ordenação ótima.



**Figura 4.1. Estruturas originais das redes Bayesianas Ásia e Câncer Metastático [60].**

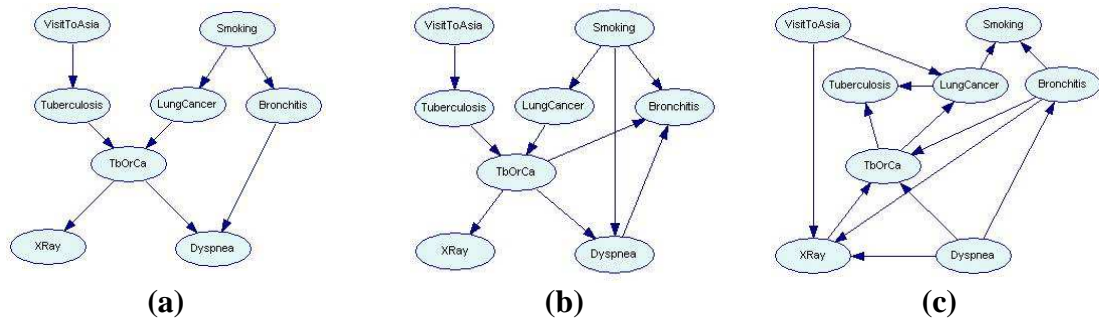
Os valores de  $g$  obtidos pelo K2 para as bases de dados *Ásia* e Câncer Metastático demonstram a sensibilidade deste algoritmo em relação à ordenação de variáveis. A Tabela 4.2 apresenta os valores de  $g$  calculados com cada uma das ordenações (ótima, intermediária e ruim), exibidas na Tabela 4.3. As estruturas aprendidas, referentes a estas ordenações, são mostradas nas Figura 4.2 e Figura 4.3. Percebe-se que a ordenação ótima produziu, para cada base, os maiores valores de  $g$  e, consequentemente, as melhores estruturas (Figura 4.2 (a) e Figura 4.3 (a)), iguais às originais (Figura 4.1). A ordenação intermediária produziu valores de  $g$  próximos dos valores obtidos com a ordenação ótima em cada base e as estruturas geradas (Figura 4.2 (b) e Figura 4.3 (b)) também ficaram próximas da original. Já a ordenação ruim produziu valores de  $g$  muito baixos em relação à ordenação ótima e as estruturas geradas (Figura 4.2 (c) e Figura 4.3 (c)) ficaram bem diferentes da original, com vários arcos a mais e vários arcos invertidos. Este experimento demonstra que a ordenação das variáveis pode influenciar diretamente no valor da função  $g$  e na correspondente estrutura aprendida.

**Tabela 4.2. Valores de  $g$  obtidos por K2, com ordenação ótima, intermediária e ruim.**

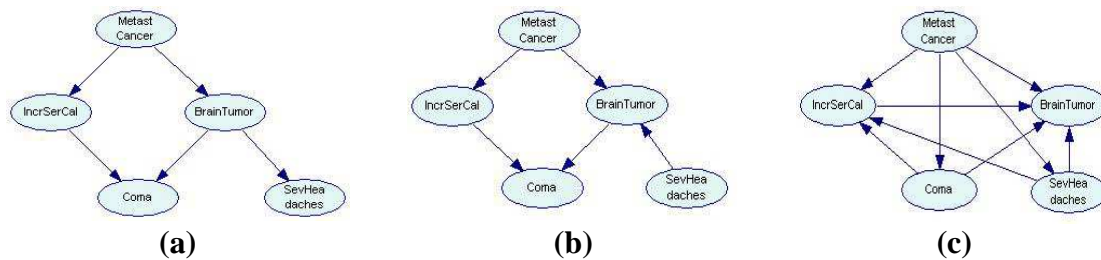
Base de Dados	Valor $g$ (K2) (Ordenação ótima)	Valor $g$ (K2) (Ordenação intermediária)	Valor $g$ (K2) (Ordenação ruim)
<i>Ásia</i>	-39740.8	-39757.2	-39802.4
Câncer Metastático	-20131.3	-20138.7	-20180.2

**Tabela 4.3. Ordenações ótima, intermediária e ruim, fornecidas ao K2 para a indução das redes.**

Base de Dados	Ordenação ótima	Ordenação intermediária	Ordenação ruim
<i>Ásia</i>	VisitToAsia, Tuberculosis, Smoking, LungCancer, TbOrca, Bronchitis, XRay, Dyspnea.	VisitToAsia, Tuberculosis, Smoking, LungCancer, TbOrca, XRay, Dyspnea, Bronchitis.	VisitToAsia, Dyspnea, Bronchitis, XRay, TbOrca, LungCancer, Smoking, Tuberculosis.
Câncer Metastático	MetastCancer, BrainTumor, IncrSerCal, Coma, SevHeadaches.	MetastCancer, IncrSerCal, SevHeadaches, BrainTumor, Coma.	MetastCancer, SevHeadaches, Coma, IncrSerCal, BrainTumor.



**Figura 4.2.** Estruturas das redes *Bayesianas Asia*, aprendidas pelo K2, com ordenação ótima (a), ordenação intermediária (b) e ordenação ruim (c).



**Figura 4.3.** Estruturas das redes *Bayesianas do Câncer Metastático*, aprendidas pelo K2, com ordenação ótima (a), ordenação intermediária (b) e ordenação ruim (c).

#### 4.2.2 Taxa de Classificação

A taxa de classificação pode ser usada como critério de avaliação da estrutura da rede *Bayesiana*, caso a rede seja usada como classificador, mas ela não sofre influência da ordenação das variáveis.

Quando a rede *Bayesiana* é utilizada como classificador, uma das variáveis da base de dados deve ser considerada como variável classe. A estrutura gerada deverá ser adequada (talvez a ótima) para a inferência desta variável. No entanto, esta estrutura pode não representar corretamente todos os relacionamentos entre as variáveis e fornecer uma boa taxa de classificação (é o caso do *Naive Bayes*) [30].

A taxa de classificação, contudo, não deve ser utilizada para refletir a influência da ordenação, pois a rede *Bayesiana* armazena a distribuição de probabilidade conjunta das variáveis e, para a tarefa de classificação, a ordem ou a direção das variáveis não tem a mesma influência. Isto pode ser observado no exemplo do Câncer Metastático, onde as taxas de classificação obtidas pelas estruturas aprendidas com ordenações diferentes foram iguais (veja a Tabela 4.4). Este fato motiva a hipótese de que, quando

se deseja apenas a classificação por *ranking* (onde a precisão da estimativa de probabilidades não é importante), não há razões fortes para que se busque a melhor ordenação das variáveis antes da indução do classificador. Por este motivo, para tal situação foram desenvolvidos, neste trabalho, dois algoritmos (DMBC e A-DMBC) de indução rápida de classificadores *Bayesianos*, mostrados no Capítulo 5.

**Tabela 4.4. Taxas de classificação obtidas por K2, com ordenação ótima, intermediária e ruim.**

Base de Dados	K2 (Ordenação ótima)	K2 (Ordenação intermediária)	K2 (Ordenação ruim)
<i>Asia</i>	51.3	51.3	51.13
<b>Câncer Metastático</b>	82.36	82.36	82.36

### 4.2.3 Estimativas de Probabilidades

As estimativas de probabilidades são referentes às de probabilidades associadas à classe mais provável de uma determinada instância do conjunto de dados. Elas são importantes quando as saídas da classificação não são usadas somente como *ranking*, mas combinadas com outras fontes de informações para a tomada de decisão. Por exemplo, conhecer as estimativas de probabilidades da variável classe pode fornecer mais informações sobre custos de predições incorretas (mesmo se os custos associados não estão exatamente definidos) [61].

Uma maneira simples de se obter uma medida sobre a precisão das estimativas de probabilidades é calculando-se a distância Euclidiana entre as probabilidades originais (verdadeiras) – dadas pelo modelo original da rede *Bayesiana* – e as probabilidades estimadas pelo modelo aprendido por algum algoritmo, como o K2, por exemplo. Esta medida não é diretamente proporcional à “qualidade” da ordenação e será chamada precisão da estimativa de probabilidade, ou PEP. Observe, por exemplo, a Tabela 4.5, a qual apresenta os resultados das PEPs obtidas com modelos gerados com o K2 para as bases de dados *Asia* e Câncer Metastático, utilizando ordenação ótima, intermediária e ruim. Nota-se que, para a base de dados do Câncer, à medida que a ordenação se distancia da ordenação ótima, os valores das probabilidades estimadas também se distanciam das probabilidades verdadeiras. Esta monotonicidade indicaria que a ordenação das variáveis influencia de maneira diretamente proporcional no resultado das estimativas de probabilidades. Mas, para a base de dados *Asia*, isto não é verdade, pois a ordenação ótima e a ordenação intermediária produziram as mesmas

estimativas de probabilidades, não refletindo sua influência de maneira diretamente proporcional. Ainda assim, nota-se nestes exemplos uma tendência de que melhores ordenações tragam bons resultados nas estimativas de probabilidade.

**Tabela 4.5.** Distância entre os valores das probabilidades verdadeiras e os valores das probabilidades estimadas pelo K2, com ordenação ótima, intermediária e ruim.

Base de Dados	K2 (Ordenação ótima)	K2 (Ordenação intermediária)	K2 (Ordenação ruim)
<i>Asia</i>	7573.33	7573.33	11505.98
<i>Câncer Metastático</i>	53.99	71.68	128.32

Existem, também, outros métodos capazes de obter estimativas de probabilidades a partir de pontuações do classificador, como proposto em [62] e [63].

### 4.3 Busca da Ordenação usando Estratégias Evolucionárias

Como já mencionado, a informação da ordenação de variáveis adequada pode não estar disponível em aplicações do mundo real [46]. Encontrar uma boa ordenação é uma questão importante e não há métodos computacionais exatos e eficientes para encontrá-la. Assim algumas heurísticas têm sido usadas [46].

Alguns autores têm usado mecanismos de busca sequencial, como descrito na Subseção 2.6.1, enquanto outros utilizam técnicas de otimização evolucionária, das quais algoritmos genéticos (GAs) são bastante empregados. Estas técnicas evolucionárias têm sido o método de escolha para muitos problemas difíceis [39].

Em [47], GAs desenvolvidos para o problema do caixeiro viajante (*Traveling Salesman Problem*) são aplicados para encontrar uma ordenação de variáveis adequada ao aprendizado de redes *Bayesianas*.

Em [64], dois tipos de Estimativas de Algoritmos de Distribuição – EDAs – (UMDA e MIMIC), com codificação discreta e contínua, são usados para buscar pela melhor ordenação para o algoritmo K2. EDAs são um novo paradigma para Computação Evolucionária que têm sido usado como ferramenta de busca no problema de aprendizado de estrutura de redes *Bayesianas*.

*Hsu et al.* [65] desenvolveram um GA com base no operador de cruzamento de ordem (OX) que usa o K2 para o aprendizado de estrutura e uma medida de aptidão flexível com base na perda inferencial.

*Kabli et al.* [66] introduziram o “*chainGA*” para evoluir uma população de ordenações topológicas de nós de redes *Bayesianas*. Em cada passo de avaliação, uma estrutura de cadeia da ordenação dada é construída e avaliada usando a métrica K2. As ordenações então assumem a aptidão retornada pelas cadeias associadas e são evoluídas, cruzadas e mutadas para um número predefinido de avaliações. No fim de cada execução, o algoritmo de busca gulosa K2 é executado sobre uma porcentagem das melhores ordenações encontradas para buscar pela melhor estrutura de rede.

*Falkner* [67] apresentou um algoritmo chamado K2GA: um algoritmo genético que usa uma versão modificada da heurística K2 para melhorar a eficiência da busca.

Em [68] e [69], foram propostos dois métodos híbridos, que aplicam a teoria de algoritmos evolucionários e aprendizado de redes *Bayesianas*, chamados de: VOGA (*Variable Ordering Genetic Algorithm*) [68] e VOGAC (*Variable Ordering Genetic Algorithm using Classification*) [69]. VOGA utiliza um algoritmo genético para codificar as possíveis ordenações de variáveis do domínio em cromossomos. A função de aptidão, utilizada por VOGA para avaliar os cromossomos, foi definida em função da pontuação *Bayesiana* calculada pelos algoritmos de aprendizado de redes *Bayesianas*, baseados em busca heurística. VOGAC diferencia-se de VOGA na definição da função de aptidão. Ao invés de usar a pontuação *Bayesiana* para avaliar os cromossomos, VOGAC utiliza a Taxa de Classificação Correta (TCC) obtida pelos classificadores gerados a partir das ordenações contidas nos cromossomos. Dessa forma, VOGAC pode empregar algoritmos de aprendizado, baseados no conceito de independência condicional.

#### 4.4 Considerações Finais

A informação da ordenação das variáveis é uma restrição importante que pode facilitar bastante o aprendizado de redes *Bayesianas*. Infelizmente, gerar uma boa ordenação exige uma quantidade significativa de conhecimento do domínio, o que não é disponível em muitas aplicações práticas.

Os trabalhos apresentados nas subseções anteriores propõem métodos para resolver o problema de busca da ordenação de variáveis, visando a otimização do aprendizado de redes *Bayesianas*. Em cada um destes trabalhos, encontram-se referências a métodos anteriores e descrevem-se melhorias que poderiam ser feitas ou



problemas que precisam ser corrigidos, o que demonstra que encontrar a ordenação ótima ainda é um desafio.

O problema da busca de uma ordenação adequada das variáveis foi previamente investigado durante o projeto de mestrado, onde foi proposta a aplicação de algoritmos evolucionários, com operadores genéticos clássicos, para se obter tal ordenação [70]. Os resultados obtidos incentivaram o desenvolvimento desta tese, visando-a compreender a influência da ordenação de variáveis no processo de indução da estrutura de redes *Bayesianas* e propor novas abordagens com o objetivo de se obter resultados mais eficientes.

Nesta tese, o entendimento deste problema foi aprofundado e identificaram-se casos onde a ordenação de variáveis pode ser necessária ou não. Desta forma, foram desenvolvidos métodos de aprendizado de estruturas que se aplicam a estes casos.

No Capítulo 5, são apresentados dois métodos (DMBC e A-DMBC) que não dependem da ordenação de variáveis para a indução de estruturas de redes *Bayesianas* utilizadas na tarefa de classificação. Estes métodos podem ser vistos como uma alternativa quando não se deseja gastar um tempo elevado na busca pela melhor ordenação.

No Capítulo 6, são apresentadas estratégias evolucionárias para buscar pela ordenação de variáveis adequada à otimização do aprendizado de estruturas. Neste caso, o uso destes métodos implica em um esforço computacional mais alto, mas pode ser mais adequado quando o interesse é uma identificação mais provável das relações de dependência entre as variáveis. Quando as relações entre as variáveis do domínio são conhecidas, informações relevantes podem ser obtidas e usadas para a tomada de decisões.

## 5 Aprendizado de Estruturas com DMBC e A-DMBC

Algoritmos desenvolvidos para induzir redes *Bayesianas* irrestritas podem ser usados também para induzir Classificadores *Bayesianos*, como tem sido feito em [2][71][72]. Contudo, é necessário sobrepor às dificuldades em termos de complexidade computacional, impondo restrições adicionais que geram ganhos extras de eficiência computacional. Normalmente, tais restrições são especificadas com foco na precisão de classificação, sem levar em consideração outros aspectos importantes para um bom modelo de classificação, tais como a precisão de estimativas de probabilidades [73] e as relações entre as variáveis [11].

Como declarado em [74], um classificador geralmente é uma parte de um grande processo de decisão, no qual as estimativas exatas de probabilidades, assim como o conhecimento das variáveis relevantes, fornecem utilidade adicional. Por exemplo, conhecendo a estimativa da probabilidade da classe, pode-se dar mais informação sobre custos de predições incorretas (mesmo se os custos associados não são precisamente definidos) [61].

Quando se deseja apenas a classificação por *ranking* (onde a precisão da estimativa de probabilidades não é importante) não há razões fortes para que se busque a melhor ordenação antes da indução do classificador. Pois, para a tarefa de classificação, a ordem ou a direção das variáveis não tem a mesma influência. Por este motivo, desenvolveu-se neste trabalho um novo método com o objetivo de induzir classificadores *Bayesianos* precisos, tendo estimativas de probabilidades de confiança e revelando relações atuais entre as variáveis mais relevantes. Os algoritmos propostos são chamados de DMBC e A-DMBC.

### 5.1 DMBC: *Dynamic Markov Blanket Classifier*

*Dynamic Markov Blanket Classifier* (DMBC) é um algoritmo desenvolvido a partir do K2 para aprender redes *Bayesianas* especialmente projetadas para problemas de classificação. A idéia de induzir classificadores usando a função  $g$  do K2 (função (2.3)) tem como objetivo reduzir o peso introduzido pelas suposições de independência embutidas no classificador *Naive Bayes*, melhorando assim as estimativas de probabilidade, enquanto mantém boas taxas de classificação e, também, permite a

identificação de relações importantes entre as variáveis mais relevantes. Além disso, como o algoritmo tende a reduzir o número de variáveis da rede gerada, o problema da ordenação de variáveis pode ser minimizado. Desta forma, o DMBC pode ser visto como uma alternativa quando não se deseja gastar um tempo elevado na busca pela melhor ordenação.

O algoritmo DMBC é basicamente uma versão do algoritmo K2 projetado para explorar a cobertura de *Markov* da variável classe (CMB), de forma que a indução da estrutura fosse mais rápida e produzisse estimativas mais exatas das probabilidades da classe. A idéia principal desta abordagem é excluir, das possíveis estruturas (DAG), aquelas que têm variáveis que não pertencem ao CMB.

A Figura 5.1 resume o DMBC em uma forma algorítmica. É possível notar que DBMC diferencia-se de K2 (Figura 2.5) nas linhas 10, 11, 15 e 16. As linhas 10 e 11 servem para testar se um nó  $z$ , que precede o nó  $i$  atual na lista ordenada, está presente no CMB. Isto é uma restrição extra imposta pelo DMBC. Assim, todas as variáveis identificadas como membros do CMB não são testadas e não são incluídas como pais de outras variáveis, permitindo uma redução do número de possíveis estruturas a serem investigadas. As linhas 15 e 16 verificam se o nó atual é filho da classe e, caso não seja, interrompe a busca de outros pais para este nó. Como o DMBC é baseado em um método heurístico, não há garantias que o CMB encontrado seja composto por todos os nós presentes na cobertura de *Markov* do nó classe (CMB\*).

É importante observar que o DMBC busca por uma estrutura que mapeia as relações entre a classe e as outras variáveis no conjunto de dados. Portanto, apesar de usar o conceito de cobertura de *Markov*, DMBC pode incluir variáveis fora do CMB\* na estrutura aprendida. Como declarado em [30], o tamanho do CMB\* pode, de certo modo, influenciar os resultados da classificação. Neste sentido, classificadores induzidos com um pequeno CMB\* tendem a ser piores que classificadores simples como NB, enquanto que o CMB construído por DMBC pode melhorar o desempenho da classificação. Além disso, relações entre as variáveis são também permitidas pelo algoritmo DMBC.

## Algoritmo DMBC

{Entrada: conjunto de nós  $n$ , uma ordenação dos nós (tendo a classe como primeiro nó), um limite  $u$  sobre o número de pais que um nó pode ter e uma base de dados  $D$  contendo  $m$  casos.}

{Saída: Para cada nó, a identificação dos pais do nó.}

```

1.  CMB := {}; / Cobertura de Markov da Classe /
2.  for  $i := 1$  to  $n$  do
3.       $\pi_i := \{\}$ ;
4.       $P_{old} := g(i, \pi_i)$ ; / equação (2.3) /
5.      OKToProceed := true;
6.      while OKToProceed and  $\pi_i < u$  do
7.          let  $z$  be the node in  $Pred(x_i) - \{\pi_i \cup CMB\}$  that
              maximizes  $g(i, \pi_i \cup \{z\})$ ;
8.           $P_{new} = g(i, \pi_i \cup \{z\})$ ;
9.          if  $P_{new} > P_{old}$  then
10.             if ( $z$  is parent of any class child) or
                    ( $z$  is a class child) then
11.                  $CMB := CMB \cup \{z\}$ ;
12.                  $P_{old} := P_{new}$ ;
13.                  $\pi_i := \pi_i \cup \{z\}$ ;
14.             end {if};
15.             if (not_class_child( $i$ ) and class  $\notin \pi_i$ ) then
16.                 break_while();
17.             else OKToProceed := false;
18.         end {while};
19.         write('Node:',  $x_i$ , 'Parents of this node:',  $\pi_i$ )
20.     end {for};
21. end {DMBC};

```

**Figura 5.1. Pseudocódigo do algoritmo DMBC.**

## 5.2 A-DMBC: Approximate Dynamic Markov Blanket Classifier

Outro algoritmo proposto neste trabalho é o *Approximate Dynamic Markov Blanket Classifier* (A-DMBC). Este algoritmo usa as mesmas idéias apresentadas para o DMBC. A única diferença entre o DMBC e sua versão simplificada está relacionada ao CMB. No A-DMBC, em vez de usar o CMB, outro conceito (veja Definição 5.1) é usado para criar uma cobertura aproximada de *Markov* da classe (*Approximate Class' Markov Blanket* – ACMB).

*Definição 5.1.* Suponha uma rede *Bayesiana*  $G$ . Para cada variável  $X$  em  $G$ , o conjunto de todos os pais de  $X$  e filhos de  $X$ , é uma cobertura de *Markov* aproximada de  $X$  ( $AMB(X)$ ).

*Definição 5.2.* Considere  $G$  um classificador *Bayesiano*. Todos os nós em  $G$  presentes na cobertura de *Markov* aproximada do nó classe formam a cobertura de *Markov* aproximada do nó classe e é denotada por ACMB\*.

*Observação.* A cobertura de *Markov* aproximada gerada pelo algoritmo A-DMBC é denotada como ACMB.

A cobertura de *Markov* aproximada foi criada com base na seguinte suposição: em um procedimento de propagação de crença na rede *Bayesiana*, quanto mais distante um nó está da evidência, menos ele é influenciado pela evidência. Tal suposição torna possível imaginar que os nós mais próximos do nó classe têm mais influência na crença da classe.

É importante mencionar que o ACMB é uma solução heurística aplicada para reduzir o esforço computacional necessário pelo algoritmo de indução do classificador *Bayesiano*. Assim, baseado na função  $g$  (função (2.3)), ele induz um simples classificador a partir de dados, tendo apenas a variável classe, seus pais e seus filhos. Como pode ser visto na Figura 5.2, em sua implementação, considera-se sempre que a variável classe é um nó raiz (um nó não tendo pais) no classificador. Por isso, o classificador *Bayesiano* induzido pelo A-DMBC tende ser mais próximo ao classificador *Naive Bayes* que o classificador induzido pelo DMBC. Também o A-DMBC não permite a identificação de relações entre todas as variáveis no CMB\*.

**Algoritmo A-DMBC**

Entrada: conjunto de nós  $n$ , uma ordenação dos nós (tendo a classe como primeiro nó), um limite  $u$  sobre o número de pais que um nó pode ter e uma base de dados  $D$  contendo  $m$  casos.

{Saída: Para cada nó, a identificação dos pais do nó.}

```

1. ACMB := {}; / Approximate Class' Markov Blanket /
2. for  $i := 1$  to  $n$  do
3.    $\pi_i := \{\}$ ;
4.    $P_{old} := g(i, \pi_i)$ ; / equation (2.3) /
5.    $P_{new} = g(i, \{class\_node\})$ ;
6.   if  $P_{new} > P_{old}$  then
7.      $\pi_i := \pi_i \cup \{class\_node\}$ ;
8.     ACMB := ACMB  $\cup \{i\}$ ;
9.   end {if}
10.  write('Node:',  $x_i$ , 'Parents of this node:',  $\pi_i$ )
11. end {for};
12. end {A-DMBC};
```

**Figura 5.2.** Pseudocódigo do algoritmo A-DMBC.

## 5.3 Avaliação Experimental

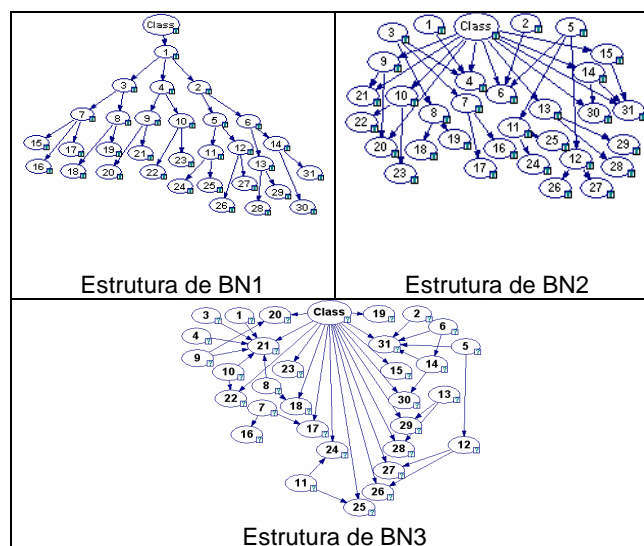
Esta seção descreve as configurações experimentais adotadas. A seguir, são apresentados e analisados os resultados obtidos com os algoritmos avaliados.

### 5.3.1 Configuração Experimental

Os experimentos são baseados no desejo de avaliar o desempenho relativo dos algoritmos *Bayesianos* estudados sob condições controladas. Com este propósito em mente, foram concebidos três cenários experimentais particulares. Para cada cenário, foram construídos modelos conceituais de redes *Bayesianas* que codificam as distribuições de probabilidade conjunta sobre um conjunto de variáveis aleatórias. Estes modelos conceituais reproduzem domínios hipotéticos de interesse. Tais modelos foram então usados para gerar conjunto de dados sintéticos (SDs), formados por 5000 instâncias. Cada um destes conjuntos de dados pode ser visto como uma realização particular de uma dada distribuição de probabilidade que representa o respectivo modelo conceitual de rede *Bayesiana*. Desta forma, distribuições de probabilidade verdadeiras são conhecimentos *a priori* para cada conjunto de dados usados nos experimentos, permitindo derivar análises interessantes a respeito do desempenho relativo dos algoritmos sob investigação.

O primeiro cenário tem o objetivo de reproduzir domínios que apresentam algumas variáveis interdependentes. Para este cenário, foram construídos três modelos sintéticos de redes *Bayesianas* (BN1, BN2 e BN3). As estruturas destes modelos são descritas na Figura 5.3 (os nomes das variáveis são representados por números nos nós de cada grafo).

BN1 representa um domínio descrito por 32 variáveis, incluindo a classe. A Figura 5.3 mostra que apenas uma variável influencia diretamente a variável classe. Em outras palavras, uma única variável faz parte da cobertura de *Markov* (CMB) de BN1. Além disso, todas as variáveis têm ao menos um pai. Portanto, BN1 representa uma poliárvore. Em aplicações do mundo real, poliárvores dificilmente são adequadas para modelar a distribuição de probabilidade das variáveis [1]. Estas árvores são, contudo, estruturas adequadas para estudar o desempenho de classificadores em domínios onde as inter-relações entre variáveis são consideradas simples.



**Figura 5.3. Estruturas de redes Bayesianas para o primeiro cenário experimental.**

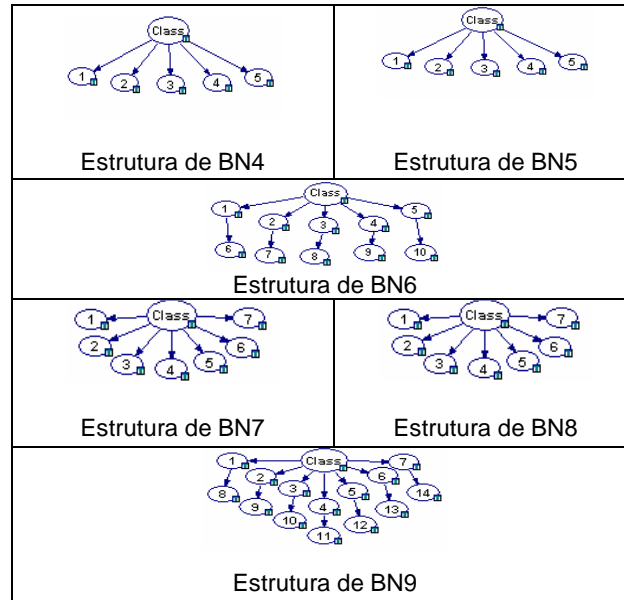
Similarmente à BN1, BN2 (Figura 5.3) tem 32 variáveis. Diferentemente de BN1, entretanto, em BN2 14 variáveis influenciam diretamente a variável classe. Além disso, cada variável de BN2 pode ter três pais (no máximo), levando assim a relações de interdependência mais complexas entre as variáveis que aquelas encontradas em estruturas de poliárvores, tais como as variáveis representadas por BN1. Portanto, BN2 é um modelo menos restritivo que BN1, potencialmente representando problemas de mineração de dados práticos mais comuns.

O primeiro cenário experimental é concluído com a descrição da BN3, a qual também contém 32 variáveis. Todas estas variáveis fazem parte do CMB, e cada uma delas pode ter sete pais (no máximo).

Para cada modelo de rede *Bayesiana* descrito, um conjunto de dados é gerado e referenciado aqui como SD1, SD2 e SD3, fazendo referência à BN1, BN2 e BN3, respectivamente. Como um comentário geral, é possível dizer que estes conjuntos de dados (principalmente SD2 e SD3) tendem a favorecer aos classificadores de redes *Bayesianas* irrestritas (tais como os classificadores induzidos pelo K2).

O segundo cenário tem como objetivo reproduzir domínios onde o viés indutivo do *Naive Bayes* é particularmente apropriado. Assim, as variáveis nas redes *Bayesianas* construídas para este cenário são condicionalmente independentes, dado a classe. Como descrito em [75], “não é realístico esperar que esta suposição seja mantida no mundo natural. Correlações entre as variáveis em um dado domínio são comuns”. Contudo,

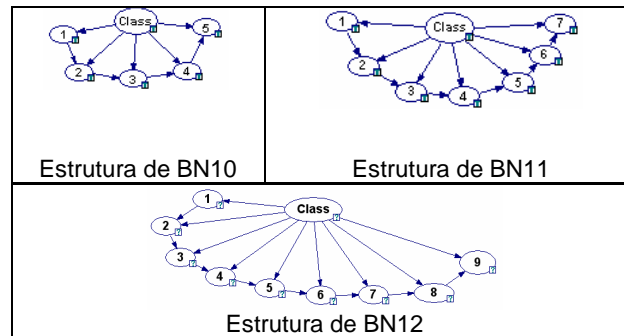
projetar experimentos para simular domínios onde o classificador *Naive Bayes* é favorecido pode esclarecer sobre o desempenho relativo dos outros classificadores *Bayesianos* sendo aqui investigados. Dois modelos (BN4 e BN7, veja Figura 5.4) simulam tal cenário. Além disso, duas variações de cada um destes modelos de redes foram criadas, uma com a adição de variáveis irrelevantes (BN5) e outra com a inclusão de variáveis (BN6). Particularmente, BN5 é baseada em BN4, cujos parâmetros numéricos foram modificados para converter as variáveis 1, 2 e 3 em variáveis irrelevantes. BN6 é obtida pela inclusão de cinco variáveis novas e redundantes em BN4, chamadas de variáveis: 6, 7, 8, 9 e 10. Analogamente à derivação de BN5 e BN6 de BN4, BN8 e BN9 foram concebidas de BN7 (veja Figura 5.4). Mais precisamente, BN8 tem cinco variáveis irrelevantes (1, 2, 3, 4, e 5), enquanto BN9 tem sete variáveis redundantes (8, 9, 10, 11, 12, 13 e 14). Visto que variáveis redundantes podem impactar o desempenho de *Naive Bayes*, o modelo BN9 pode ser visto, em princípio, como um modelo não muito favorável para classificadores *Naive Bayes*. Contudo, os resultados experimentais, que são informados na seção 5.2.2, mostram que as variáveis redundantes não têm impactado significativamente o desempenho de *Naive Bayes*.



**Figura 5.4. Estruturas de redes Bayesianas para o segundo cenário experimental.**

Finalmente, o terceiro cenário é formado por três modelos de redes Bayesianas (BN10, BN11 e BN12) que são baseadas no “Tree Augmented Networks (TAN)” [30]. Estas estruturas são descritas na Figura 5.5.





**Figura 5.5. Estruturas de redes Bayesianas para o terceiro cenário experimental.**

Em resumo, para cada modelo de rede descrito, um conjunto de dados foi gerado. O primeiro conjunto de dados sintéticos (SD1) é formado por 5000 instâncias, as quais podem ser vistas como uma realização particular da distribuição de probabilidade representada por BN1. Similarmente, são obtidos SD2 de BN2 e assim por diante. A Tabela 5.1 resume as características principais de cada conjunto de dados usados nestes experimentos. A SubSeção 5.3.2 procede com os resultados obtidos.

**Tabela 5.1. Conjuntos de dados usados nos experimentos: número de atributos (AT), número de instâncias (IN) e número de valores da classe (NV).**

	SD1	SD2	SD3	SD4	SD5	SD6	SD7	SD8	SD9	SD10	SD11	SD12
#AT	31	31	31	5	5	10	7	7	14	5	7	9
#IN	5k	5k	5k	5k	5k	5k	5k	5k	5k	5k	5k	5k
#NV	2	2	2	2	2	2	2	2	2	2	2	2

### 5.3.2 Experimentos e Análises de Resultados

Nesta seção, são apresentados os resultados do desempenho dos algoritmos K2, DMBC, A-DMBC, NB e TAN, usando os conjuntos de dados descritos na Subseção 5.3.1 como entrada. Inicialmente, é informado o número de chamadas à função  $g$  – equação (2.3) – para induzir os classificadores Bayesianos que dependem desta função (K2, DMBC e A-DMBC). Os resultados são mostrados na Tabela 5.2 e na Figura 5.6. Considerando que os experimentos executados usaram conjuntos de dados de estruturas conhecidas, a ordenação “ótima” foi usada em todas as induções conduzidas.

**Tabela 5.2. Número de chamadas à função g.**

Base de Dados	K2	DMBC	A-DMBC
SD1	987.5	82.9	63.0
SD2	1059.8	179.0	63.0
SD3	1082.0	309.6	63.0
SD4	31.0	11.0	11.0
SD5	28.0	17.0	11.0
SD6	111.0	21.0	21.0
SD7	53.0	15.8	15.0
SD8	46.2	25.0	15.0
SD9	196.5	48.2	29.0
SD10	36.4	11.0	11.0
SD11	62.3	16.0	15.0
SD12	103.6	34.9	19.0
<b>Rank (Friedman)</b>	<b>3.00</b>	<b>1.88</b>	<b>1.13</b>

Como esperado, a Tabela 5.2 e o gráfico descrito na Figura 5.6 mostram que DMBC e A-DMBC trouxeram melhorias significativas sobre K2 para todas as bases de dados, quando o número de chamadas à função g (o qual é o núcleo de DMBC e K2) é levado em consideração. O número de vezes que DMBC e A-DMBC chamaram a função g é, em média, menos que 23% e 9% de vezes que a função g foi chamada pelo algoritmo K2, respectivamente. A Figura 5.6 destaca o fato de que os domínios de aplicação, tendo número maior de variáveis e número maior de relações entre as variáveis (primeiro cenário no experimento), tendem a gerar maior esforço computacional quando usando o K2 e o DMBC. A-DMBC, por sua vez, tem seu esforço computacional baseado no número de variáveis do problema. Assim, é constante em SD1, SD2 e SD3. A complexidade assintótica do K2 (como implementado neste experimento) é  $O(m \ 100 \ n^2 \ r)$ , enquanto DMBC tem sua complexidade assintótica dada por  $O(m \ 100 \ (n-/CMB/) \ n \ r)$  e A-DMBC por  $O(m \ r \ 2(n-1))$ .

A Tabela 5.2 e a Figura 5.6 também permitem empiricamente avaliar a magnitude dos termos constantes – negligenciados pela complexidade de tempo assintótico. Como o TAN não é baseado na função g, seus resultados não são informados. É importante mencionar, contudo, que o tempo de complexidade do TAN [30] é  $O(v^2 \ n)$  onde  $v$  é o número de vértices no final da estrutura TAN e  $n$  é o número de variáveis.

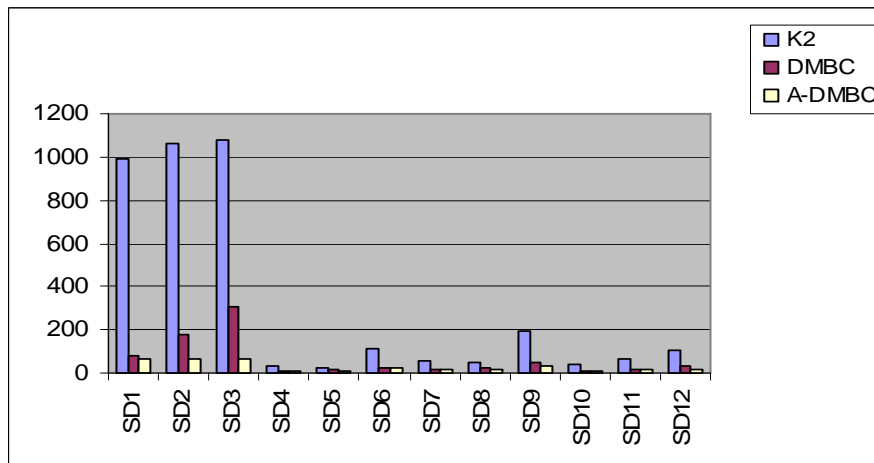


Figura 5.6. Número de avaliações da função g.

Em geral, para fornecer uma imagem melhor do desempenho relativo dos algoritmos sob estudo, são informados os resultados de comparações estatísticas. Seguindo *Demsar* [76], foi aplicado o teste estatístico de *Friedman* e, quando aplicável, os testes *post-hoc* de *Nemenyi* para avaliar os resultados obtidos. Sob a hipótese nula, que declara que todos os algoritmos são equivalentes, tem-se  $p = 0.000011$ , assim, a hipótese nula é rejeitada (ou seja, para  $\alpha=5\%$ ) e procede-se com os testes *post-hoc* de *Nemenyi*. Neste sentido, DMBC e A-DMBC mostraram melhor desempenho que K2. Embora os *ranks* usados no cálculo do teste de *Friedman* sugeriram que A-DMBC é computacionalmente mais eficiente que DMBC, tal diferença observada não é estatisticamente significativa em  $\alpha=5\%$ .

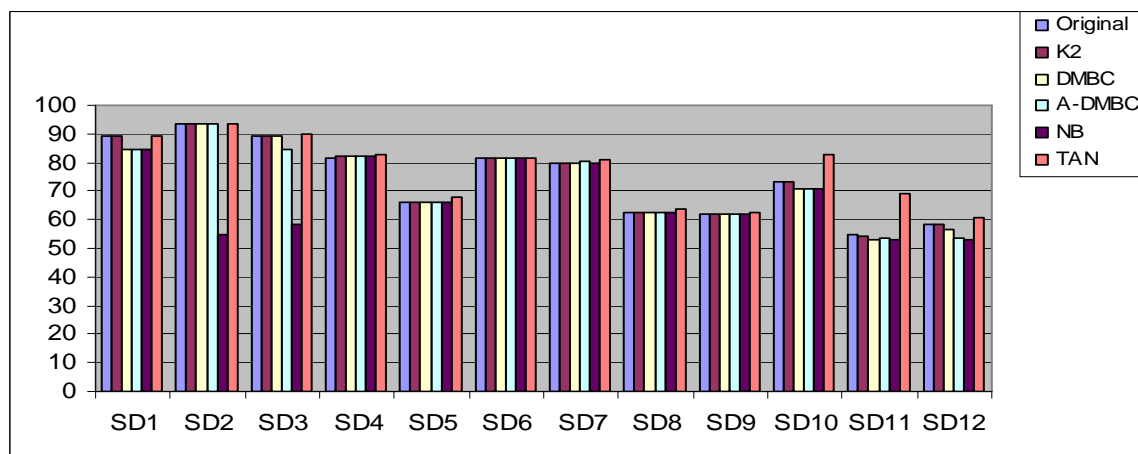
A capacidade de generalização de cada algoritmo foi estimada usando uma estratégia de validação cruzada *10-fold*. Os mesmos conjuntos de treinamento e teste foram usados para avaliar a precisão de classificação dos algoritmos. As taxas médias de classificação correta (TMCC) são resumidas na Tabela 5.3 e plotados na Figura 5.7. A segunda coluna – Original – refere-se aos resultados obtidos pelas tuplas de re-classificação de cada conjunto de dados de acordo como o modelo de rede *Bayesiana* original que o gerou.

Falando a grosso modo, muitos dos classificadores aplicados mostraram desempenho similar sobre os 12 conjuntos de dados usados nos experimentos. Contudo, assumindo que todos os algoritmos têm TMCC equivalentes, um valor  $p$  de  $0.000035$  foi obtido a partir do teste de *Friedman*, sugerindo que a hipótese nula fosse rejeitada (ou seja, para  $\alpha=5\%$ ) e que procedesse com os testes *post-hoc* de *Nemenyi*. Estes testes mostraram que apenas o desempenho de TAN é significativamente diferente dos outros

classificadores nestes conjuntos. É importante mencionar, contudo, que a diferença crítica entre dois *ranks* médios para o teste de *Nemenyi* é igual a 2.17, que é precisamente a diferença entre TAN e A-DMBC (veja a última linha da Tabela 5.3). Considerando a coluna original (veja a Tabela 5.3, segunda coluna), como a descrição correta de um classificador ótimo, quanto mais próximo o valor “*Rank de Friedman*” é de 3.54 (veja Tabela 5.3, última linha), melhor é o desempenho de classificação.

**Tabela 5.3. Taxas médias de classificação correta (TMCC) obtidas por validação cruzada 10-fold.**

Base de Dados	Original	K2	DMBC	A-DMBC	NB	TAN
SD1	89.14	89.14	84.23	84.31	84.23	89.41
SD2	93.41	93.38	93.38	93.38	54.67	93.52
SD3	89.10	89.20	89.12	84.35	58.24	90.15
SD4	81.77	81.85	81.85	82.05	81.85	82.46
SD5	66.20	66.20	66.20	66.33	66.20	67.79
SD6	81.41	81.41	81.41	81.58	81.41	81.62
SD7	79.93	79.92	79.92	80.07	80.05	81.01
SD8	62.53	62.53	62.53	62.66	62.53	63.52
SD9	61.61	61.61	61.61	61.74	61.61	62.72
SD10	73.16	72.97	70.79	70.94	70.79	82.82
SD11	54.91	54.05	53.03	53.43	52.75	68.86
SD12	58.12	58.08	56.66	53.64	53.21	60.56
<b>Rank (Friedman)</b>	<b>3.54</b>	<b>3.83</b>	<b>4.46</b>	<b>3.17</b>	<b>5.00</b>	<b>1.00</b>



**Figura 5.7. Taxas médias de classificação correta em uma estratégia de validação cruzada 10-fold.**

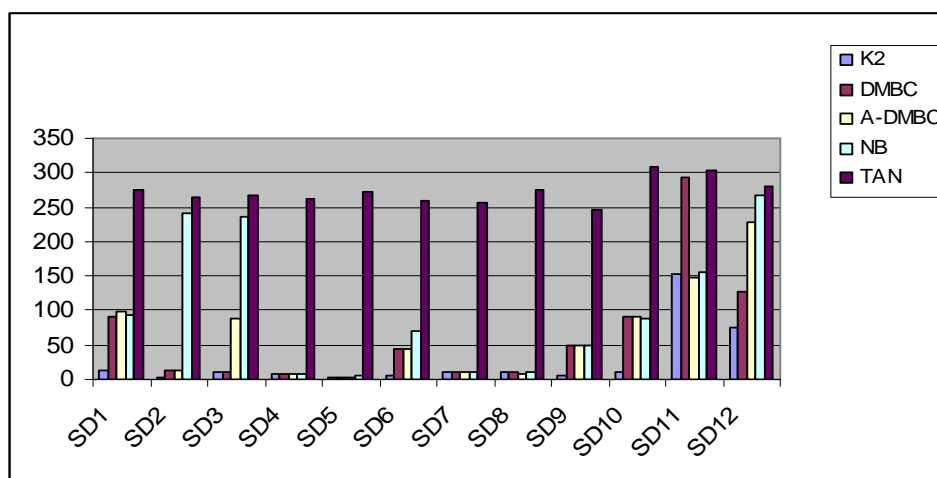
A Figura 5.7 mostra que no primeiro cenário DMBC e A-DMBC apresentaram o comportamento esperado. O desempenho de DMBC mostrou-se ser mais próximo de K2 (exceto para o SD1) e A-DMBC executou levemente pior que K2 em SD1 e SD3. Também, como esperado, *Naive Bayes* mostrou bom desempenho no segundo cenário e TAN alcançou altas TMCCs no terceiro cenário. Em geral, no segundo cenário, todos os algoritmos comportaram-se de forma muito parecida e, no terceiro cenário, TAN tendeu

a ser favorecido e apresentou boas TMCCs não-realísticas quando comparado às redes originais.

Finalmente, o interesse principal está em comparar o desempenho de cada algoritmo na tarefa de estimar as estimativas de distribuição de probabilidade da classe, as quais são *a priori* conhecidas para cada conjunto de dados usado nestes experimentos. Para isto, foram usadas duas métricas de distância, chamadas: a distância Euclidiana e a divergência *Kullback-Leibler* (KL) entre as probabilidades originais, dadas pelos modelos de RB usadas para gerar os conjuntos de dados, e aquelas estimativas de probabilidades, dadas por cada algoritmo sobre os conjuntos de testes gerados no processo de validação cruzada. Os resultados obtidos são resumidos na Tabela 5.4 e Figura 5.8 para a distância Euclidiana e na Tabela 5.5 e Figura 5.9 para a divergência KL.

**Tabela 5.4. Distância Euclidiana entre os valores de probabilidades verdadeiros e os valores de probabilidades estimados pelos algoritmos.**

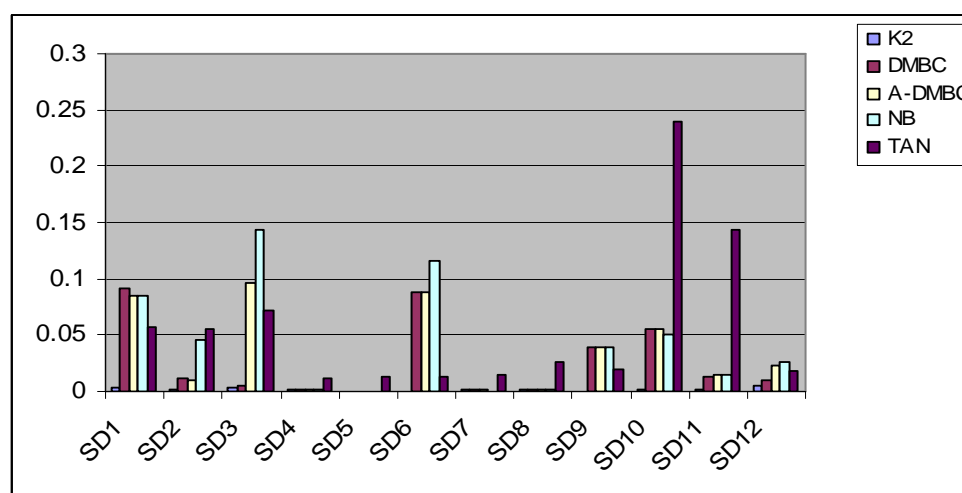
Base de Dados	K2	DMBC	A-DMBC	NB	TAN
SD1	13.36	91.77	97.82	92.77	273.83
SD2	3.66	14.06	11.85	240.95	264.92
SD3	9.43	10.45	88.18	237.13	267.94
SD4	6.73	6.73	6.83	6.76	262.67
SD5	2.13	2.13	2.25	6.18	270.94
SD6	4.02	43.88	43.87	69.65	260.40
SD7	10.34	10.34	10.38	9.64	257.29
SD8	11.42	11.42	8.15	10.50	274.14
SD9	5.50	50.46	50.43	50.20	247.54
SD10	9.36	91.58	91.69	89.32	309.25
SD11	151.83	291.86	148.58	156.83	304.25
SD12	75.46	126.08	229.30	266.12	278.87
<b>Rank (Friedman)</b>	<b>1.50</b>	<b>2.67</b>	<b>2.83</b>	<b>3.00</b>	<b>5.00</b>



**Figura 5.8. Distância euclidiana entre os valores de probabilidade verdadeiros e aqueles estimados pelos algoritmos.**

**Tabela 5.5. Divergência KL entre os valores de probabilidades verdadeiros e os valores de probabilidades estimados pelos algoritmos.**

Base de Dados	K2	DMBC	A-DMBC	NB	TAN
SD1	0.00392	0.09049	0.084795	0.08559	0.05707
SD2	0.00103	0.01208	0.009097	0.04529	0.05543
SD3	0.00375	0.00436	0.095726	0.14387	0.07117
SD4	0.00115	0.00115	0.001540	0.00117	0.01173
SD5	0.00019	0.00019	0.000426	0.00074	0.01230
SD6	0.00061	0.08847	0.088429	0.11549	0.01253
SD7	0.00088	0.00088	0.00088	0.00081	0.01420
SD8	0.00233	0.00233	0.001153	0.00212	0.02538
SD9	0.00065	0.03855	0.038572	0.03976	0.01923
SD10	0.00167	0.05613	0.056112	0.05041	0.24005
SD11	0.00224	0.01244	0.014693	0.01504	0.14294
SD12	0.00507	0.00998	0.023068	0.02578	0.01793
<b>Rank (Friedman)</b>	<b>2.67</b>	<b>3.50</b>	<b>4.00</b>	<b>4.41</b>	<b>4.91</b>



**Figura 5.9. Divergência Kullback-Leibler entre os valores de probabilidade verdadeiros e aqueles estimados pelos algoritmos.**

Sob a hipótese nula, que declara que todos os algoritmos são equivalentes, tem-se  $p = 0.0000028$  para a distância Euclidiana e  $p = 0.0011$  para a divergência KL, assim a hipótese nula é rejeitada (ou seja, para  $\alpha=5\%$ ) e procede-se com os testes *post-hoc* de *Nemenyi*. Estes testes mostram que apenas TAN tem fornecido um desempenho significativamente pior que os outros algoritmos – a diferença crítica entre dois *ranks* médios é 2.17. Isto parece estar em conformidade com os resultados apresentados na Tabela 5.3, cujos resultados sugerem algum *overfitting* pelo classificador induzido por TAN. Nenhuma diferença estatística foi observada nas comparações par a par entre os outros classificadores. Como esperado, K2 forneceu estimativas de probabilidades melhores na maioria dos conjuntos de dados. Dado que SD2 e SD3 foram gerados a

partir dos modelos mais gerais de RB, e que tais conjuntos de dados podem ser considerados mais adequados para descrever domínios de mineração de dados do mundo real (por exemplo, veja [1] para detalhes), os experimentos executados, de alguma forma, confirmam porque algoritmos baseados no K2 são geralmente preferidos quando a precisão de estimativas de probabilidades é de suprema importância. Mesmo no segundo cenário (SD4, ..., SD9), que favorecem o *Naive Bayes*, pode ser observado que algoritmos baseados no K2 fornecem boas estimativas de probabilidades.

### 5.3.3 Discussão

Neste capítulo, foi descrito um algoritmo chamado *Dynamic Markov Blanket Classifier* (DMBC) e uma variação aproximada deste algoritmo, que foi denominada de A-DMBC. O objetivo destes algoritmos é induzir classificadores *Bayesianos* a partir de dados. Estes algoritmos foram projetados com a motivação de contornar algumas limitações indesejáveis dos classificadores *Bayesianos* mais usados, como *Naive Bayes*, e, ao mesmo tempo, melhorar o tempo computacional necessário para aprender um Classificador de Rede *Bayesiana*. Os algoritmos propostos foram avaliados através de experimentos executados em 12 conjuntos de dados. Comparações com os algoritmos TAN e *Naive Bayes* foram realizadas. As principais conclusões derivadas dos experimentos são: (i) A-DMBC mostrou ser mais eficiente computacionalmente que seus homólogos baseados no K2 (e mais eficiente que o TAN, já que sua complexidade é menor); (ii) em muitos conjuntos de dados, as taxas médias de classificação correta (TMCC) são próximas às alcançadas pelo TAN, que mostrou alcançar as mais altas TMCC neste estudo; (iii) em muitos conjuntos de dados, o desempenho de A-DMBC na estimação das distribuições de probabilidades é melhor que TAN e *Naive Bayes*, mas é pior que K2.

Considerando todas estas razões, é possível acreditar que DMBC e A-DMBC fornecem um bom equilíbrio entre eficiência computacional e precisão na tarefa de induzir classificadores *Bayesianos* a partir de dados. Portanto, quando o classificador a ser induzido for usado como parte de um processo de decisão maior, para o qual as estimativas de probabilidades precisas, assim como o conhecimento sobre as variáveis relevantes fornecem utilidade adicional, os métodos propostos tendem a ser mais

adequados para induzir classificadores de redes *Bayesianas* que K2, TAN e *Naive Bayes*.



## 6 Métodos Evolucionários de Busca da Ordenação

### 6.1 Considerações Iniciais

O problema de encontrar a rede *Bayesiana* que melhor reflete as relações de dependência de uma base de dados é difícil por causa do grande número de estruturas DAG possíveis, até mesmo para um número pequeno de nós conectados. A metodologia padrão para abordar este problema é a execução de buscas heurísticas sobre algum espaço de busca.

A ordenação das variáveis do problema tem sido uma restrição comum em algoritmos de aprendizado de redes *Bayesianas* como forma de reduzir o espaço de busca do processo de aprendizado [5] ( $n!$  em vez de  $2^{O(n^2)}$  [77]). Como visto no Capítulo 4, é possível encontrar na literatura algoritmos propostos que realizam uma busca no espaço de ordenações e não no espaço de estruturas, selecionando para cada ordenação a melhor rede consistente com ela.

No projeto de mestrado [70], desenvolvido anteriormente, foram empregados algoritmos evolucionários para realizar a busca de uma ordenação de variáveis adequada, visando à otimização do aprendizado de redes *Bayesianas* para a tarefa de classificação. Os métodos híbridos propostos, chamados de VOGA (*Variable Ordering Genetic Algorithm*) [68], VOGAC (*Variable Ordering Genetic Algorithm using Classification*) [69] e VOEA (*Variable Ordering Evolutionary Algorithm*) [70], obtiveram bons resultados e foram capazes de encontrar boas ordenações de variáveis.

As características dos métodos híbridos, apresentados em [70], motivaram uma investigação mais profunda do problema de busca da ordenação de variáveis. O Capítulo 5 apresentou dois métodos desenvolvidos para aprender redes *Bayesianas* para a tarefa de classificação. Para este tipo de problema, a ordenação das variáveis não reflete sua influência. No entanto, para os casos onde a ordenação das variáveis é indicada, esta tese apresenta novas estratégias evolucionárias com o objetivo de buscar a ordenação adequada para otimizar este aprendizado.

## 6.2 Métricas de Pontuação utilizadas como Funções de Aptidão

Vários autores têm proposto o uso de Algoritmos Evolucionários (AEs) para encontrar uma ordenação adequada, visando, em geral, à otimização do aprendizado de estruturas de redes *Bayesianas*. Em [70], foi proposto um algoritmo genético, (o qual é um AE) chamado VOGA, que busca por tal ordenação para induzir estruturas de redes *Bayesianas*.

VOGA usou uma métrica de pontuação (pontuação *Bayesiana* – função  $g$  (2.3)), definido no algoritmo K2, como sua função de aptidão. A métrica de pontuação calcula a probabilidade conjunta de uma estrutura de rede *Bayesiana* e uma determinada base de dados, representando uma medida de qualidade que demonstra o quanto a estrutura de rede é adequada, como uma representação própria para um dado conjunto de dados. Portanto, a métrica de pontuação representa uma função crucial em métodos *Bayesianos*, como o K2 [78].

Pesquisadores têm proposto uma variedade de métricas de pontuação com base em diferentes suposições para induzir uma rede *Bayesiana* a partir de dados. A métrica que traz melhores resultados torna-se mais interessante [78]. Já que VOGA foi implementado utilizando uma métrica de pontuação como função de aptidão, são avaliadas, neste trabalho, outras quatro métricas que também poderiam ser usadas por VOGA para este fim: Entropia [79], AIC (*Akaike Information Criterion*) [80], BDe (*Bayesian Dirichlet equivalent*) [18] e MDL (*Minimum Description Length*) [27].

O principal objetivo deste propósito é avaliar o desempenho de VOGA com diferentes métricas de pontuação e verificar qual delas (ou uma possível combinação delas) pode trazer melhores resultados.

As Subseções 6.2.1 a 6.2.4 apresentam as métricas avaliadas neste trabalho. Nestas subseções, as seguintes convenções são utilizadas:

- i)  $r_i$  ( $1 \leq i \leq n$ ) é a cardinalidade do atributo  $x_i$  ( $i=1, \dots, n$ );
- ii)  $q_i$  denota a cardinalidade do conjunto de pais do atributo  $x_i$  na estrutura de rede  $B_s$ , ou seja, o número de valores diferentes para os quais os pais de  $x_i$  ( $pa(x_i)$ ) podem ser instanciados. Assim,  $q_i$  pode ser calculado como o produto de cardinalidades dos nós em  $pa(x_i)$ ,  $q_i = \prod_{x_j \in pa(x_i)} r_j$  (Note que  $pa(x_i)=\emptyset$  implica  $q_i=1$ );

- iii)  $N_{ij}$  ( $1 \leq i \leq n$ ,  $1 \leq j \leq q_i$ ) denota o número de registros na base de dados  $D$  para o qual  $pa(x_i)$  pega seu  $j$ -ésimo valor;
- iv)  $N_{ijk}$  ( $1 \leq i \leq n$ ,  $1 \leq j \leq q_i$ ,  $1 \leq k \leq r_i$ ) denota o número de registros em  $D$  para o qual  $pa(x_i)$  pega seu  $j$ -ésimo valor e para o qual  $x_i$  pega seu  $k$ -ésimo valor. Assim,  $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ .
- v)  $N$  denota o número de registros em  $D$ .

### 6.2.1 A Métrica de Entropia

A Entropia é uma medida não negativa de incerteza, que é máxima quando a incerteza é máxima e zero quando há um conhecimento completo. Quanto mais informação é fornecida, mais baixa é a Entropia.

Segundo [79], a métrica de Entropia  $H(B_s, D)$  de uma estrutura de rede e uma base de dados pode ser definida como:

$$H(B_s, D) = -N \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}} \quad (6.1)$$

e o número de parâmetros  $K$  pode ser definido como:

$$K = \sum_{i=1}^n (r_i - 1) \cdot q_i \quad (6.2)$$

### 6.2.2 A Métrica AIC

*Akaike* [80] introduziu o critério de informação para seleção de modelos, generalizando seu trabalho anterior sobre análises de séries de tempo e análises de fator. A métrica AIC (*Akaike Information Criterion*)  $Q_{AIC}(B_s, D)$  de uma estrutura de rede *Bayesiana*  $B_s$  para uma base de dados  $D$  é:

$$Q_{AIC}(B_s, D) = H(B_s, D) + K \quad (6.3)$$

### 6.2.3 A Métrica BDe

A idéia básica da abordagem *Bayesiana* é maximizar a probabilidade da estrutura da rede de acordo os dados, ou seja, maximizar  $P(B_s/D)$  sobre todas as

estruturas possíveis  $B_S$ , dados os casos da base de dados  $D$ . Para este fim, a probabilidade é calculada para várias estruturas de redes e aquela estrutura com a mais alta probabilidade é selecionada.

A métrica *Bayesiana* de uma estrutura de rede  $B_S$  para uma base de dados  $D$  é:

$$Q_{Bayes}(B_S, D) = P(B_S) \prod_{i=0}^n \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \quad (6.4)$$

Onde  $P(B_S)$  é a *priori* da estrutura da rede e  $\Gamma(.)$  é a função gama.  $N'_{ij}$  e  $N'_{ijk}$  representam escolhas de *prioris* sobre contas restritas por  $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$ . Com  $N'_{ijk}=1$  (e assim  $N'_{ij}=r_i$ ), é obtida a métrica K2 (função (2.3)). Com  $N'_{ijk}=1/r_i \cdot q_i$  (e assim  $N'_{ij}=1/q_i$ ), é obtida a métrica BDe [18][79].

#### 6.2.4 A Métrica MDL

O princípio MDL (*Minimum Description Length*) surgiu da teoria de codificação onde o objetivo é encontrar uma descrição de uma base de dados tão curta quanto possível com o mínimo de parâmetros. A métrica MDL pode ser considerada como uma aproximação da medida *Bayesiana*, e assim tem uma interpretação *Bayesiana*. Contudo, ela oferece várias vantagens sobre a medida *Bayesiana* [27].

A métrica MDL  $Q_{MDL}(B_S, D)$  de uma estrutura de rede *Bayesiana* para uma base de dados  $D$  é definida como:

$$Q_{MDL}(B_S, D) = H(B_S, D) + \frac{K}{2} \log N \quad (6.5)$$

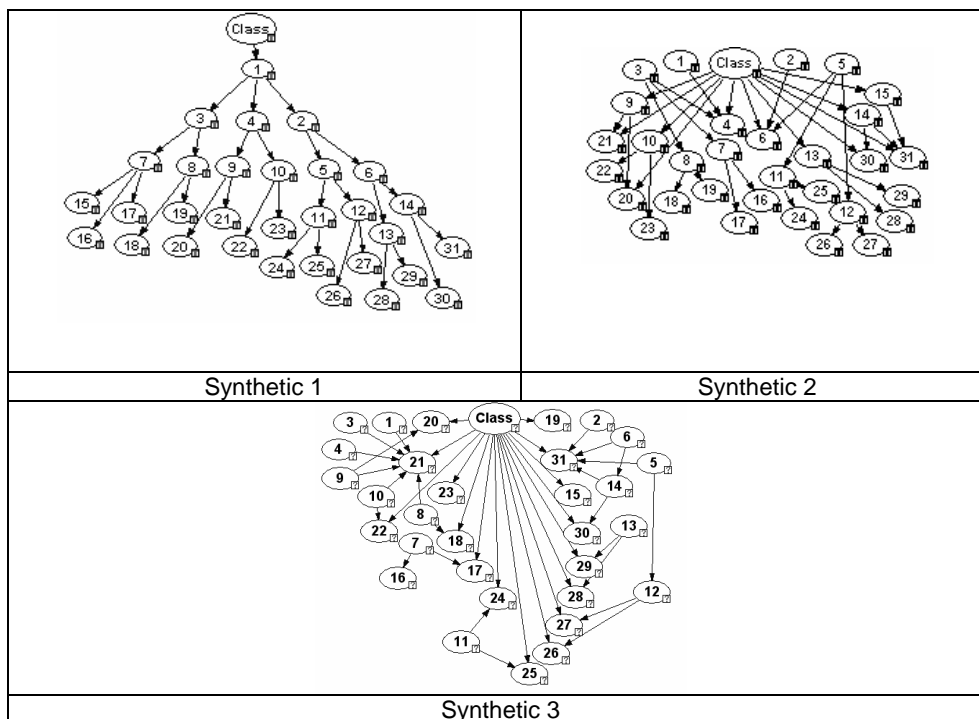
#### 6.2.5 Experimentos e Análises de Resultados

Nos experimentos apresentados nesta subseção, o algoritmo VOGA foi implementado usando diferentes métricas de pontuação como função de aptidão, tais como: função  $g$  (função (2.3)), MDL, AIC, BDe e Entropia. Além de utilizar estas medidas, foram feitas combinações entre elas de duas formas, com o objetivo de gerar duas novas funções de aptidão: i) utilizando a soma de todas estas métricas para avaliar um indivíduo; ii) utilizando os valores de todas estas métricas como votos para avaliar cada indivíduo e a estratégia de voto da maioria é aplicada.

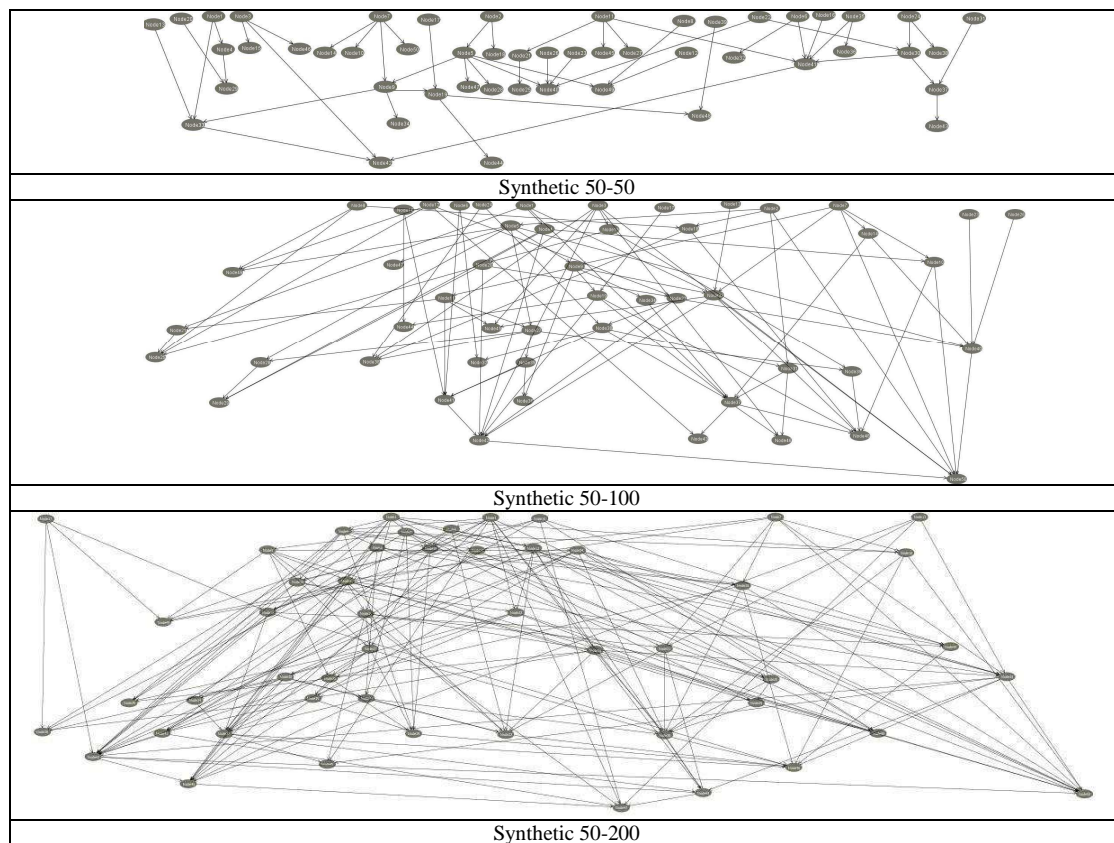
Os experimentos executados por VOGA usando diferentes métricas de pontuação envolveram seis domínios de conhecimento, os quais têm suas características resumidas na Tabela 6.1. Os domínios *Synthetic 1*, *Synthetic 2* e *Synthetic 3* possuem 32 variáveis cada e foram criados usando uma estratégia de amostra aplicada às estruturas dos classificadores *Bayesianos* descritos na Figura 6.1. Os domínios *Synthetic 50-50*, *Synthetic 50-100* e *Synthetic 50-200* possuem 50 variáveis cada, assim como 50, 100 e 200 arcos, respectivamente. A estrutura destas três redes *Bayesianas* são mostradas na Figura 6.2. O número diferente de arcos foi gerado para verificar o impacto da densidade da rede na tarefa de obter ordenações ótimas. Todos os casos de testes são binários, embora o mesmo comportamento seja esperado, caso as classes sejam maiores que 2.

**Tabela 6.1. Descrição dos conjuntos de dados, constando o nome do conjunto, número de atributos mais a classe (AT), número de instâncias (IN) e número de valores da classe (NV).**

	<i>Synthetic 1</i>	<i>Synthetic 2</i>	<i>Synthetic 3</i>	<i>Synthetic 50-50</i>	<i>Synthetic 50-100</i>	<i>Synthetic 50-200</i>
<b>AT</b>	32	32	32	50	50	50
<b>IN</b>	10000	10000	10000	50000	50000	50000
<b>NV</b>	2	2	2	2	2	2



**Figura 6.1. Redes Bayesianas representando os domínios *Synthetic 1*, *Synthetic 2* e *Synthetic 3*. As representações gráficas foram criadas usando o software GeNie [60].**



**Figura 6.2.** Redes *Bayesianas* representando os domínios *Synthetic 50-50*, *Synthetic 50-100* e *Synthetic 50-200*. As representações gráficas foram criadas usando o software *Weka* [81].

Os seguintes passos definem a metodologia experimental para cada um dos seis conjuntos de dados em mais detalhes:

1. Os experimentos envolveram VOGA usando os operadores clássicos de cruzamento (OX) e de mutação (mutação com base em ordem), como descrito em [70].
2. A taxa de cruzamento foi definida com o valor de 0.8 e a taxa de mutação com valor de 0.3. Estes valores foram definidos empiricamente.
3. Para cada conjunto de dados, todos os algoritmos foram avaliados em um número de execuções, usando a mesma população, a qual tem 100 indivíduos.
4. Cada conjunto de dados foi executado pelas versões de VOGA usando as cinco métricas de pontuação como função de aptidão (função  $g$ , Entropia, AIC, BDe e MDL). A Tabela 6.2 apresenta uma comparação entre as estruturas de rede induzidas por VOGA e as estruturas originais, informando o número de arcos extras, o número de arcos ausentes e o número de arcos invertidos.
5. As duas outras versões de VOGA foram implementadas usando uma combinação das cinco métricas como função de aptidão. A primeira estratégia de combinação

avalia os indivíduos através do voto da maioria. Isto é, cada indivíduo (ordenação) é avaliado pelas cinco métricas. O indivíduo tendo os maiores valores de pontuação (para cada métrica) terão mais votos e serão considerados as melhores escolhas. Este algoritmo é chamado de “VOGA – Voto da maioria”. A segunda versão usa a soma de todos as pontuações como função de aptidão e é chamada de “VOGA – Soma das pontuações”. A Tabela 6.2 também mostra as diferenças entre as estruturas originais e as induzidas por estas duas versões de VOGA.

6. Cada execução do algoritmo termina depois de 10 gerações sem melhorias e retorna a melhor ordenação encontrada, assim como a rede *Bayesiana* correspondente. A Tabela 6.3 apresenta o número de gerações necessárias até a convergência.

**Tabela 6.2. Número de arcos extras (+), número de arcos ausentes (-) e número de arcos invertidos (R).**

	VOGA-g			VOGA-Entropia			VOGA-MDL			VOGA-AIC			VOGA-BDe			“VOGA-Soma das pontuações”			“VOGA-Voto da maioria”		
	+	-	R	+	-	R	+	-	R	+	-	R	+	-	R	+	-	R	+	-	R
<b>Synth 1</b>	<b>3</b>	<b>3</b>	<b>2</b>	8	7	1	3	2	4	5	1	6	6	1	7	6	3	6	7	3	7
<b>Synth 2</b>	<b>3</b>	<b>8</b>	<b>5</b>	8	8	6	<b>3</b>	<b>8</b>	<b>5</b>	4	8	4	<b>3</b>	<b>8</b>	<b>5</b>	<b>3</b>	<b>8</b>	<b>5</b>	<b>3</b>	<b>8</b>	<b>5</b>
<b>Synth 3</b>	<b>1</b>	<b>14</b>	<b>1</b>	4	14	3	0	14	4	<b>0</b>	<b>14</b>	<b>2</b>	1	15	3	1	14	2	6	14	8
<b>Synth 50-50</b>	18	1	18	23	2	18	25	2	19	15	1	14	<b>12</b>	<b>1</b>	<b>12</b>	18	2	14	22	2	19
<b>Synth 50-100</b>	66	3	24	84	5	34	<b>55</b>	<b>5</b>	<b>28</b>	84	5	34	70	2	29	85	3	32	84	5	34
<b>Synth 50-200</b>	<b>299</b>	<b>76</b>	<b>38</b>	305	83	34	304	78	42	306	70	41	305	84	44	304	75	41	311	81	49

A Tabela 6.2 exibe o número de arcos extras, arcos ausentes e arcos invertidos das redes induzidas pelas versões de VOGA em comparação com as redes originais. É possível perceber que VOGA usando a função *g* como função de aptidão tende a produzir estruturas mais próximas das originais. Esta versão foi a melhor nos domínios *Synthetic 1*, *Synthetic 2* e *Synthetic 3* e *Synthetic 50-200*. VOGA-MDL produziu a mesma estrutura de rede induzida por VOGA-g quando aplicado a *Synthetic 2* e induziu a melhor estrutura quando aplicada a *Synthetic 50-100*. VOGA-AIC induziu uma estrutura muito similar à induzida por VOGA-g quando aplicada a *Synthetic 3*. VOGA usando as combinações das métricas obteve estruturas levemente diferentes das outras versões. Contudo, a Tabela 6.3 mostra que “VOGA – Voto da maioria” convergiu mais rápido que as outras versões.

É possível notar que todas as versões de VOGA induziram estruturas muito diferentes das originais quando aplicadas a *Synthetic 50-100* e *Synthetic 50-200*. Isto

demonstra o quanto é difícil aprender estruturas de redes muito conectadas e tendo muitas variáveis. Apesar disto, a Tabela 6.4 mostra que todas as versões de VOGA obtiveram porcentagens de classificação equivalentes para todos as bases de dados, demonstrando que as diferentes métricas de pontuação têm influência similar nas porcentagens de classificação.

**Tabela 6.3. Número de gerações necessárias à convergência.**

	VOGA-g	VOGA-Entropia	VOGA-MDL	VOGA-AIC	VOGA-BDe	“VOGA-Soma das pontuações”	“VOGA-Voto da maioria”
<b>Synth 1</b>	26	22	20	24	18	23	<b>15</b>
<b>Synth 2</b>	<b>11</b>	27	<b>11</b>	14	<b>11</b>	<b>11</b>	<b>11</b>
<b>Synth 3</b>	21	26	14	15	16	27	<b>11</b>
<b>Synth 50-50</b>	20	18	20	13	15	19	<b>12</b>
<b>Synth 50-100</b>	12	<b>11</b>	12	<b>11</b>	16	17	<b>11</b>
<b>Synth 50-200</b>	21	23	20	21	26	19	<b>13</b>

**Tabela 6.4. Porcentagens de classificação obtidas pelas versões de VOGA.**

	VOGA-g	VOGA-Entropia	VOGA-MDL	VOGA-AIC	VOGA-BDe	“VOGA-Soma das pontuações”	“VOGA-Voto da maioria”
<b>Synth 1</b>	89.67	89.67	89.67	89.67	89.67	89.67	89.67
<b>Synth 2</b>	93.4	93.4	93.4	93.4	93.4	93.4	93.4
<b>Synth 3</b>	89.44	89.44	89.44	89.44	88.56	89.44	<b>89.52</b>
<b>Synth 50-50</b>	69.56	69.56	69.56	69.56	69.56	69.56	69.56
<b>Synth 50-100</b>	83.65	<b>83.72</b>	83.62	<b>83.72</b>	83.65	83.56	<b>83.72</b>
<b>Synth 50-200</b>	98.88	<b>98.98</b>	98.72	98.69	98.78	98.86	98.97

## 6.2.6 Discussão

Neste trabalho, inicialmente foram avaliadas algumas métricas de pontuação utilizadas para avaliar estruturas de redes *Bayesianas*. Estas métricas são conhecidas como uma medida de qualidade que pode ser usada para estimar a probabilidade da estrutura da rede, sendo um mapeamento próprio de uma dada base de dados.

O objetivo principal da avaliação destas métricas é verificar a influência causada por elas no desempenho do algoritmo VOGA, o qual é um algoritmo genético. Os resultados empíricos mostraram que VOGA usando a função  $g$  (função (2.3)), definida pelo algoritmo K2, tende a produzir estruturas de redes mais próximas das originais.



Portanto, esta função será adotada como função de aptidão para os novos algoritmos evolucionários apresentados nas subseções seguintes.

### 6.3 Operador de Mutação com base em Distância (DMO)

Neste trabalho, é proposto um novo operador de mutação, chamado de mutação com base em distância (*Distance-based Mutation Operator – DMO*). DMO recebe três variáveis como entrada: i) a primeira é o cromossomo alvo  $c$ , ii) a segunda é o gene alvo  $g$ , iii) e a terceira é a distância  $d$  (veja Figura 6.3). Desta forma, com base na distância  $d$ , DMO seleciona aleatoriamente um segundo gene  $g'$  ( $g \neq g'$ ) em  $c$  e executa a permutação usando os valores  $g$  e  $g'$ . Para ser mais específico, suponha um cromossomo  $c$  tendo  $n$  genes representados como inteiros em  $[1, 2, 3, \dots, n]$ . Ao executar a operação de mutação usando DMO, suponha que o gene escolhido a ser modificado seja o quarto gene e a distância seja igual a 2 ( $g = 4$  e  $d = 2$ ). Então,  $g'$  deve ser selecionado entre os genes 2 e 6, incluindo estes.

#### Algoritmo DMO

```
{Entrada: cromossomo  $c$ , gene alvo  $g$ , distance  $d$ .}
{Saída:  $c'$ }
1. intervalo = um conjunto de valores de  $d-g$  a  $d+g$ 
2.  $g' = \text{random}(\text{intervalo})$ 
3.  $c' = \text{permutacao}(c, g, g')$  /*  $c'$  é o novo cromossomo onde  $g$  e  $g'$  foram permutados. */
4. end{DMO}
```

Figura 6.3. Algoritmo DMO.

A idéia inicial do operador DMO é manter exploração e exploração do algoritmo evolutivo e alcançar maior eficiência sem usar o operador de cruzamento. Assim, a busca poderá ser direcionada conforme a distância de permutação entre dois genes. Se a distância for grande, a permutação dos genes pode ser considerada global. Ou seja, podem ser gerados novos cromossomos onde a variável que era a última (na lista ordenada) pode até se tornar a primeira. Por consequência, pode drasticamente alterar os testes pai/filho executados pelo K2. Por outro lado, se a distância for pequena, a permutação tende a ser local e gerar cromossomos mais próximos à população média, sem desviar bruscamente da região de busca explorada.

O operador DMO vasculha o espaço de busca e procura pela ordenação adequada de uma forma dinâmica. Uma simples permutação de genes mais próximos pode levar a busca para uma outra área do espaço de busca, iniciando a busca em uma nova região, sem drasticamente mudar a geração. Assim, é possível buscar por ordenações adequadas dinamicamente evitando ótimos locais.

Nos experimentos executados na Subseção 6.3.1, a distância é alterada durante o curso de gerações. A distância diminui de acordo com a evolução dos melhores indivíduos, de tal forma que os bons cromossomos não sofram mudanças abruptas nas últimas gerações.

A distância pode ser definida dinamicamente e, inicialmente, ela é igual ao tamanho do cromossomo e vai decrescendo ao longo das gerações. Ela é calculada usando a seguinte função:

$$t_{atual} = t_{Max} * \frac{(k_{Max} - k_{Medio})}{(k_{Max} - k_{Min})} \quad (6.6)$$

onde  $t_{atual}$  é o tamanho atual da distância entre dois genes do cromossomo,  $t_{Max}$  é o tamanho máximo da distância entre dois genes (inicialmente igual ao tamanho do cromossomo),  $k_{Max}$  é a distância máxima entre as ordenações da geração atual,  $k_{Medio}$  é a distância média entre as ordenações e  $k_{Min}$  é a distância mínima entre as ordenações.

Neste trabalho, são aplicadas duas métricas para medir a distância entre duas ordenações, como descrito abaixo:

1. Distância de *Kendall* (*Kendall tau distance*) [82]: é uma métrica amplamente usada e que conta o número de pares discordantes entre duas listas. Quanto maior a distância, mais dissimilares são as duas listas. A distância de *Kendall* pode ser definida como:

$$K(\tau_1, \tau_2) = \sum_{\{i,j\} \in P} \bar{K}_{i,j}(\tau_1, \tau_2) \quad (6.7)$$

onde:

- $P$  é o conjunto de pares não ordenados de elementos distintos nas listas  $\tau_1$  e  $\tau_2$ ;
- $\bar{K}_{i,j}(\tau_1, \tau_2) = 0$ , se  $i$  e  $j$  estão na mesma ordem em  $\tau_1$  e  $\tau_2$ ;
- $\bar{K}_{i,j}(\tau_1, \tau_2) = 1$ , se  $i$  e  $j$  estão ordem oposta em  $\tau_1$  e  $\tau_2$ ;

2. Coeficiente de correlação de *rank* de *Spearman* [83]: é frequentemente referenciado como sendo o coeficiente de correlação de Pearson entre as

variáveis rankeadas. Na prática, contudo, um procedimento mais simples é normalmente usado para calcular  $\rho$ . As  $n$  pontuações  $X_i$ ,  $Y_i$  são convertidas em *ranks*  $x_i$ ,  $y_i$  e as diferenças  $d_i = x_i - y_i$  entre os *ranks* de cada observação sobre as duas variáveis são calculadas. O coeficiente de *Spearman*  $\rho$  é dado por:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (6.8)$$

A idéia de usar duas métricas diferentes para medir a distância entre as ordenações (durante a busca por ordenações de variáveis adequadas) é para identificar o quanto os resultados obtidos estão relacionados a uma métrica de distância específica.

### 6.3.1 Experimentos e Análises de Resultados

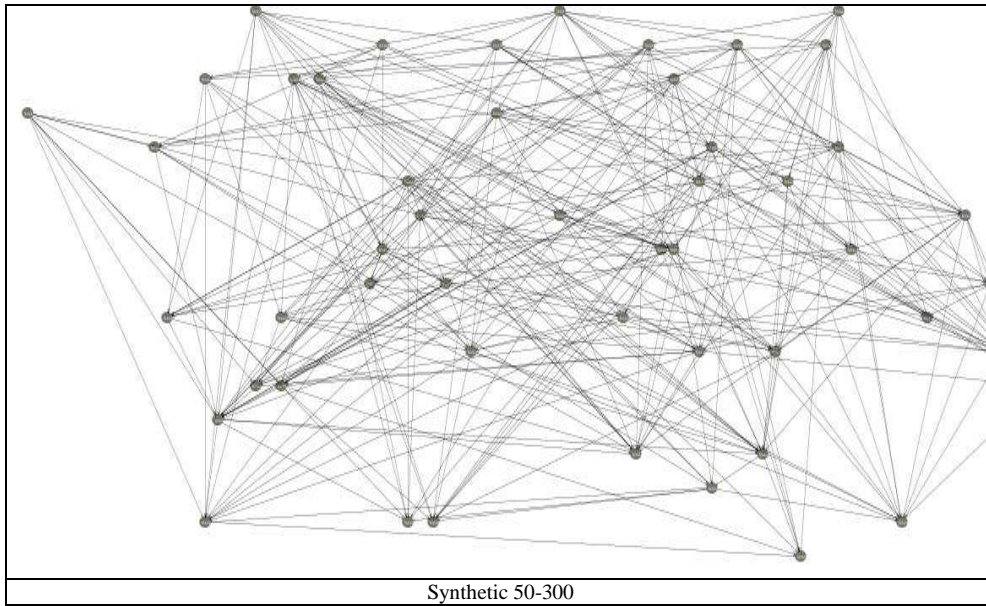
Os experimentos realizados com o operador DMO envolveram 8 domínios que têm suas características resumidas na Tabela 6.5. O domínio *Alarm* [84] é bem conhecido na comunidade de redes *Bayesianas* e possui 37 variáveis. Os domínios *Synthetic 1*, *Synthetic 2* e *Synthetic 3*, *Synthetic 50-50*, *Synthetic 50-100* e *Synthetic 50-200* são apresentados na Figura 6.1 e Figura 6.2, respectivamente, e descritos na Subseção 6.2.5. Além destes, foi acrescentado o domínio *Synthetic 50-300*, tendo 50 variáveis e 300 arcos. A estrutura desta rede é mostrada na Figura 6.4. Como já dito na Subseção 6.2.5, o número diferente de arcos foi gerado para verificar o impacto da densidade da rede na tarefa de obter ordenações ótimas.

**Tabela 6.5.** Descrição dos conjuntos de dados, constando o nome do conjunto, número de atributos mais a classe (AT), número de instâncias (IN) e número de valores da classe (NV).

	<b>Alarm</b>	<b>Synthetic 1</b>	<b>Synthetic 2</b>	<b>Synthetic 3</b>	<b>Synthetic 50-50</b>	<b>Synthetic 50-100</b>	<b>Synthetic 50-200</b>	<b>Synthetic 50-300</b>
<b>AT</b>	37	32	32	32	50	50	50	50
<b>IN</b>	10000	10000	10000	10000	50000	50000	50000	50000
<b>NV</b>	2	2	2	2	2	2	2	2

A intenção deste experimento aqui é verificar empiricamente qual o impacto de usar apenas um operador de mutação projetado especialmente para o problema de ordenação (DMO) em vez de usar operadores de crossover e mutação para permutação (como os operadores apresentados na Subseção 3.5). Para isso, cada conjunto de dados foi executado pelos algoritmos VOGA e VOGA, propostos em [70]. Estes algoritmos foram capazes de encontrar uma ordenação adequada para o aprendizado das redes

quando os autores compararam seus resultados com os resultados obtidos por K2 tendo uma ordenação ótima.



**Figura 6.4. Rede Bayesiana representando o domínios *Synthetic 50-300*. A representação gráfica foi criada usando o software *Weka* [81].**

Para estes experimentos, o algoritmo VOA foi implementado em oito versões diferentes, chamadas: VOA, VOA\_CM-f, VOA\_CM-g, VOA\_DMO-Kendall, VOA\_DMO-Spearman, VOA\_DMO-g, VOA\_DMO-Kendall-rate e VOA\_DMO-Spearman-rate. Cada versão de VOA é descrita a seguir.

1. VOA: algoritmo evolucionário utilizando o operador de mutação para permutação com base em ordem como o único operador evolucionário. Nesta versão, a taxa de mutação é mantida fixa.
2. VOA\_CM-f: algoritmo evolucionário empregando apenas o operador de mutação com base em ordem (CM). Nesta versão, o algoritmo usa taxa de mutação dinâmica, que é calculada pela função (6.9) a cada geração:

$$tx_{atual} = tx_{Max} * \frac{(g_{Max} - g_{Medio})}{(g_{Max} - g_{Min})}, \quad (6.9)$$

onde  $tx_{atual}$  é o valor da nova taxa de mutação,  $tx_{max}$  é o valor da taxa inicial fornecida pelo usuário e os valores de  $g$  referem-se às aptidões dos cromossomos da população, dadas aqui pelo cálculo da função  $g$  (função (2.3)) do K2.

3. VOA\_CM-g: esta implementação também usa um único operador de mutação clássico (CM), mas diferentemente de VOA\_CM-f, sua taxa de mutação dinâmica

é inicializada pelo usuário e diminui a cada geração. Empiricamente, foi decidido que a taxa atual seja multiplicado por 0.9 a cada nova geração, diminuindo assim a taxa de mutação geometricamente a cada geração.

4. VOEA\_DMO-Kendall: este algoritmo evolucionário implementa o operador de mutação DMO como seu único operador. A taxa de mutação é fixa e a distância de permutação de genes é dinamicamente calculada usando as funções (6.6) e (6.7).
5. VOEA\_DMO-Spearman: este algoritmo evolucionário também utiliza o operador DMO como seu único operador. A taxa de mutação é fixa e a distância de permutação de genes é dinamicamente calculada usando (6.6) e (6.8).
6. VOEA\_DMO-g: este algoritmo evolucionário utiliza DMO como seu único operador e aplica uma taxa de mutação fixa, mas diferentemente de VOEA\_DMO-Kendall e VOEA\_DMO-Spearman, a distância de permutação é inicialmente igual ao tamanho do cromossomo e, a cada geração, ela diminui. Empiricamente, decidiu-se que o tamanho da distância seja multiplicado por 0.9 em cada geração.
7. VOEA\_DMO-Kendall-rate: para estimar a influência da taxa de mutação dinâmica com o operador DMO, foi desenvolvido o VOEA\_DMO-Kendall-rate para combiná-los. Assim, VOEA\_DMO-Kendall-rate usa o operador DMO como em VOEA\_DMO-Kendall e a taxa de mutação dinâmica como em VOEA\_CM-f.
8. VOEA\_DMO-Spearman-rate: este algoritmo usa o operador DMO como em VOEA\_DMO-Spearman e a taxa de mutação dinâmica como em VOEA\_CM-f.

Os seguintes passos definem a metodologia experimental em mais detalhes para cada um dos 11 conjunto de dados:

1. Os experimentos envolveram VOGA usando os operadores de cruzamento (OX) e de mutação (mutação com base em ordem) e todas as versões de VOEA mencionadas anteriormente.
2. Para cada conjunto de dados, todos os algoritmos foram avaliados em um número de execuções, usando a mesma população.
3. A taxa de cruzamento foi definida com o valor de 0.8 e taxa de mutação com valor de 0.3. Estes valores foram definidos empiricamente.
4. A função de aptidão adotada foi a função g do K2 (função (2.3)).
5. Uma vez que os algoritmos tem natureza estocástica, mais que uma execução é necessária para verificar a solução final. Por esta razão, nos resultados apresentados neste trabalho, para cada conjunto de dados, cada algoritmo foi executado 35 vezes.

A Tabela 6.6 e a Tabela 6.7 apresentam a pontuação *Bayesiana* média (função *g*) e o número de gerações necessárias para alcançar a solução, respectivamente.

Considerando os resultados mostrados na Tabela 6.6, algumas observações são possíveis. Levando em consideração todos os domínios avaliados, as pontuações *Bayesianas* (função *g*), obtidas com as versões de VOGA, são melhores que as alcançadas por VOGA (exceto para o domínio *Synthetic* 50-50). Assim, para estes domínios, o operador de mutação sozinho tendeu a produzir as melhores ordenações de variáveis. Além disso, VOGA\_DMO-Kendall, VOGA\_DMO-Spearman, e VOGA\_DMO-Kendall-rate e VOGA\_DMO-Spearman-rate obtiveram melhores resultados que VOGA em cinco conjuntos de dados. Isto demonstra uma tendência do operador DMO encontrar melhores ordenações de variáveis que o operador clássico.

Tabela 6.6. Pontuações *Bayesianas* (função *g*).

	VOGA	VOEA	VOEA_ DMO- Kendall	VOEA_ DMO- Spearman	VOEA_ DMO-g	VOEA_ CM-f	VOEA_ CM-g	VOEA_ DMO- Kendall- rate	VOEA_ DMO- Spearman- rate
Alarm	-48349	-48092	<b>-48042</b>	-48493	-48147	-48090	-48043	-48054	-48342
Synth 1	-84823	-84794	<b>-84792</b>	-84806	-84796	<b>-84792</b>	-84794	-84793	-84797
Synth 2	-87834	<b>-87833</b>	<b>-87833</b>	-87834	<b>-87833</b>	<b>-87833</b>	<b>-87833</b>	<b>-87833</b>	<b>-87833</b>
Synth 3	-86415	-86415	-86415	-86416	-86415	-86415	-86415	-86415	-86415
Synth 50-50	<b>-1227998</b>	-1228080	-1228036	-1228126	-1228048	-1228176	-1228176	-1228176	-1228176
Synth 50-100	-1221791	-1223841	-1221224	-1220121	-1219657	-1221224	-1220579	<b>-1218931</b>	<b>-1221224</b>
Synth 50-200	-597612	-588007	-596169	-596036	-594386	-586418	-585811	<b>-578411</b>	-589208
Synth50-300	-644560	-644078	<b>-635861</b>	-644058	-639799	-644058	-642906	-636604	-640950

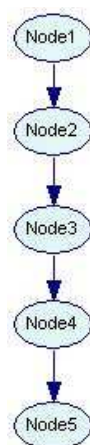
Tabela 6.7. Número de gerações necessárias para a convergência dos algoritmos VOGA e VOGA .

	VOGA	VOEA	VOEA_ DMO- Kendall	VOEA_ DMO- Spearman	VOEA_ DMO-g	VOEA_ CM-f	VOEA_ CM-g	VOEA_ DMO- Kendall- rate	VOEA_ DMO- Spearman- rate
Alarm	<b>25</b>	54	78	19	39	57	78	73	34
Synth 1	<b>26</b>	44	45	39	41	50	46	43	47
Synth 2	<b>12</b>	20	24	13	21	21	19	24	17
Synth 3	<b>25</b>	32	31	25	30	29	26	32	33
Synth 50-50	21	50	19	13	22	<b>11</b>	<b>11</b>	<b>11</b>	12
Synth 50-100	20	12	<b>11</b>	20	12	<b>11</b>	12	38	11
Synth 50-200	19	37	<b>18</b>	19	21	35	49	77	47
Synth50-300	22	15	20	12	21	11	20	26	14

A Tabela 6.7 revela que VOGA convergiu mais rápido que as versões de VOGA em 4 dos 8 conjuntos de dados. As versões de VOGA têm, contudo, alcançado melhores

valores de  $g$  para a maioria dos conjuntos de dados. Isto significa que DMO sozinho parece não permitir uma convergência mais rápida para o problema da ordenação de variáveis. Este fato pode não surpreender considerando que DMO é um operador de mutação.

Uma outra observação interessante é o fato de que DMO favoreceu estruturas de redes muito conectadas onde os nós têm muitos filhos. Estruturas de redes mais densas tendem a ter um número reduzido de ordenações ótimas. O operador DMO pode criar cromossomos mais próximos um do outro quando ele restringe o tamanho da distância de permutação entre dois genes. Por exemplo, considere os cromossomos das gerações finais da rede da Figura 6.5. Estes cromossomos não deveriam ser alterados com distância maior que 1. Porque percebe-se que a permutação de genes mais próximos de um cromossomo, como [1, 2, 3, 5, 4] (com distância de troca igual a 1), pode gerar o cromossomo [1, 2, 3, 4, 5], o qual contém a ordenação ótima para a indução da estrutura da rede da Figura 6.5.



**Figura 6.5.** Estrutura de uma rede *Bayesiana* hipotética.

### 6.3.2 Discussão

Os resultados obtidos nas análises empíricas executadas mostraram que DMO é capaz de melhorar a qualidade das ordenações de variáveis. Além disso, nos conjuntos de dados utilizados, DMO tendeu a encontrar melhores estruturas de redes *Bayesianas* (melhor valor da função  $g$ ), principalmente em domínios maiores e densamente conectados.

Pode ser visto como evidência empírica preliminar que estruturas de redes mais conectadas podem se beneficiar de usar um operador genético específico como DMO,

que limita a distância de permutação à medida que o EA segue mais distante em sua busca.

Por outro lado, o uso do operador de cruzamento e DMO pode ser uma linha de investigação interessante futuramente, principalmente para ver se tal combinação de operadores pode favorecer estruturas de redes Bayesianas tendo muitas variáveis, mas não densamente conectadas. Em estruturas de redes mais esparsas, por exemplo, a permutação de genes de um cromossomo particular pode não influenciar a geração de uma ordenação alvo. Frequentemente, pode ser melhor combiná-los com uma sequência de genes de outro cromossomo, usando operadores de cruzamento. A rede Bayesiana *Synthetic 1* (veja Figura 6.1) pode ajudar neste entendimento. É possível observar que a estrutura desta rede apresenta vários conjuntos de nós cuja ordem não é relevante. Por exemplo, o conjunto sequencial, formado a partir do “Node16” até o “Node32”, produz a mesma estrutura de rede em qualquer ordem. A mesma idéia se aplica aos nós que estão entre “Node8” e “Node16”. Portanto, para casos como estes, a combinação de sequências de genes vindas de dois cromossomos pode produzir ordenações mais interessantes que estejam mais próximas das ordenações ótimas.

## 6.4 Operador de Cruzamento de Múltiplos Pontos Aleatórios (RMX)

Neste trabalho, é proposto um novo operador de cruzamento para resolver o problema de ordenação de variáveis, chamado de operador de cruzamento de múltiplos pontos aleatórios (*Random Multi-point Crossover Operator* – RMX). RMX é adequado para modelos gráficos probabilísticos tendo arcos direcionados, como as redes Bayesianas.

RMX recebe dois cromossomos ( $c_1$  e  $c_2$ ) para serem recombinados e retorna dois novos cromossomos ( $c_3$  e  $c_4$ ) criados com base na recombinação de  $c_1$  e  $c_2$ . A Figura 6.6 descreve a idéia principal operador proposto. Primeiro, RMX configura aleatoriamente o número  $n$  de pontos de corte a ser usado. Na Figura 6.7, por exemplo,  $n = 4$ . A seguir,  $n$  posições de corte ( $p_1, p_2, \dots, p_n$ ) são aleatoriamente escolhidas. Então, os cromossomos selecionados  $c_1$  e  $c_2$  são recombinados de acordo com as posições selecionadas para gerar  $c_3$  e  $c_4$ .



**Algoritmo RMX**

```

{Entrada: cromossomos  $c_1$  e  $c_2$ }
{Saída: cromossomos  $c_3$  e  $c_4$ }
1.  $n = \text{random}(1, (\text{'número de variáveis'})/2)$  /*  $n$  é o
   número de pontos de cortes e é definido
   aleatoriamente. */
2.  $\text{posicoes} = \text{escolhePosicoesAleatorias}(n)$  /*  $n$ 
   posições de cortes são selecionadas
   aleatoriamente. */
3.  $c_3$  and  $c_4 = \text{recombinacao}(\text{posicoes}, c_1, c_2)$ 
end {RMX}

```

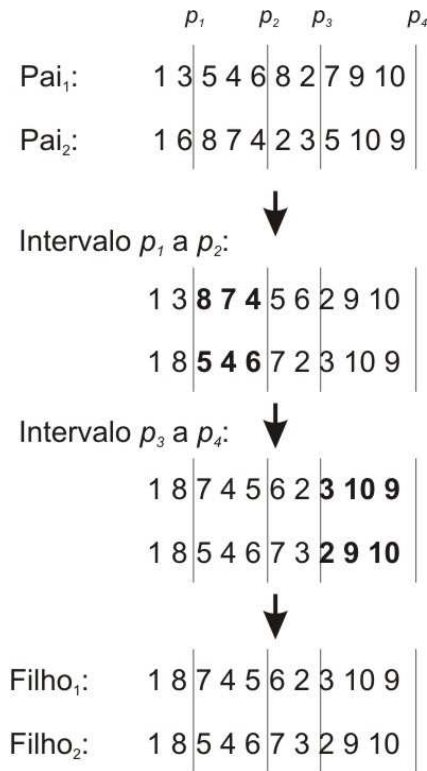
**Figura 6.6. Algoritmo RMX.**

A recombinação de genes ocorre da seguinte forma: sequências específicas de genes são trocadas alternadamente entre os dois cromossomos. Ou seja, a primeira sequência de genes (de  $p_1$  a  $p_2$ ) são trocadas entre os cromossomos; depois a terceira sequência (de  $p_3$  a  $p_4$ ); a quinta; a sétima; e assim por diante.

Para ser mais específico, observe a Figura 6.7, a qual apresenta dois cromossomos a serem recombinados:  $Parent_1$  e  $Parent_2$ . Considere que 4 pontos de corte foram aleatoriamente definidos:  $p_1, p_2, p_3$  e  $p_4$ . Os genes de  $Parent_1$ , pertencentes à sequência de  $p_1$  a  $p_2$  (5, 4, 6), são apagados de suas posições iniciais em  $Parent_2$ , deslocando os genes na sequência para a esquerda. Em seguida, “5, 4, 6” são inseridos nas posições correspondentes a  $p_1$  a  $p_2$  em  $Parent_2$ . O mesmo procedimento se aplica aos genes de  $Parent_2$  entre  $p_1$  e  $p_2$  a serem inseridos em  $Parent_1$ . A segunda sequência (de  $p_2$  a  $p_3$ ) não é utilizada. Os genes da terceira sequência são trocados como feito com a primeira sequência. No final, dois novos cromossomos filhos são gerados contendo sequências de genes de seus pais.

RMX foi projetado para ser usado como um operador único principalmente porque:

- i) Pode balancear convergência e a busca por ótimos globais (assim, não é necessário a mutação).
- ii) Favorece todos os tipos de estruturas de redes *Bayesianas*, trabalhando como um conjunto de operadores (mutação e cruzamento).



**Figura 6.7.**Aplicação do operador RMX.

RMX é um operador dinâmico e não favorece uma topologia de rede *Bayesiana* específica. Dependendo do número de pontos de corte e do tamanho das sequências selecionadas, RMX trará vantagens para diferentes topologias. Por exemplo, suponha uma rede *Bayesiana* onde todos os nós tenham 4 filhos (como mostrado na Figura 6.8 – rede *Bayesiana* Quaternária-50). A ordem entre os nós irmãos não é importante durante a indução desta estrutura. Neste caso, o que realmente importa são os pais vindo antes dos filhos na ordenação. Então, o operador RMX pode gerar a ordenação adequada para esta estrutura quando o tamanho da sequência é quatro. O operador pode transportar esses quatro nós juntos em uma ordenação ruim e colocá-los depois do pai em uma outra ordenação, produzindo a ordenação adequada. O mesmo é verdadeiro para os nós tendo diferentes números de filhos. Assim, é possível buscar por ordenações adequadas dinamicamente evitando ótimos locais.

#### 6.4.1 Experimentos e Análises de Resultados

Os experimentos executados usando RMX envolveram oito domínios, os quais têm suas características resumidas na Tabela 6.8. Em *Binary-50*, todos os nós (exceto os nós folhas que não tem filhos) têm apenas dois filhos. Em *Ternary-50*, os nós têm três

filhos. Em *Quaternary-50*, os nós têm quatro filhos. *Naive-50* tem apenas um único nó pai e todos os outros nós conectados a ele como filhos. *Chain-50* apresenta nós conectados linearmente (em uma cadeia). *NaiveChain-50* é uma mistura de *Naive-50* e *Chain-50*. Os domínios *Synthetic50-50* e *Synthetic50-100* foram gerados aleatoriamente e têm 50 variáveis cada e 50 e 100 arcos, respectivamente. O número diferente de arcos foi gerado para verificar o impacto da densidade da rede na tarefa de obter ordenações ótimas. As estruturas destas redes *Bayesianas* são mostradas na Figura 6.8. Para todos os domínios, com base em suas redes *Bayesianas* conhecidas, um conjunto de dados foi gerado usando a técnica de *Sampling-from-BN* implementada no Software *Weka* [81].

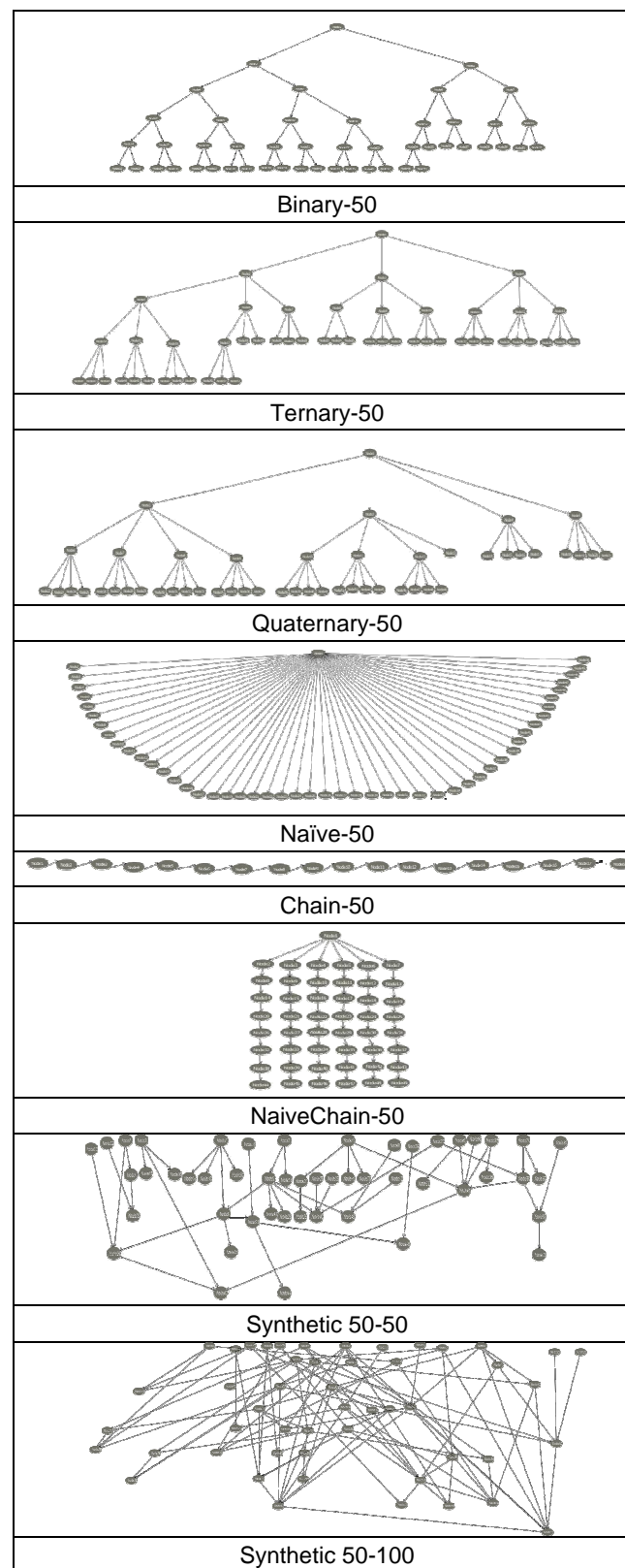
**Tabela 6.8. Descrição dos conjuntos de dados, contendo o nome do conjunto de dados (domínio), número de atributos mais a classe (AT), número de instâncias (IN) e número de arcos (AR).**

	Binary-50	Ternary-50	Quaternary-50	Naive-50	Chain-50	NaiveChain-50	Synth50-50	Synth50-100
AT	49	52	49	50	50	49	50	50
IN	10000	10000	10000	10000	10000	10000	10000	10000
AR	48	51	48	49	49	48	50	100

Nos experimentos realizados aqui, foram aplicados os algoritmos VOGA e VOGA, propostos em [70]. O algoritmo VOGA usa apenas o operador RMX e nenhum operador de mutação. A intenção é verificar empiricamente o impacto de usar o operador RMX como um único operador de cruzamento projetado especialmente. Os seguintes passos definem com mais detalhes a metodologia experimental para cada um dos oito conjuntos de dados aplicados:

1. Os experimentos envolveram VOGA usando os operadores clássicos de cruzamento (OX) e de mutação (mutação com base em ordem). Duas versões de VOGA foram implementadas. A primeira (VOGA-RMX) usa apenas RMX. A segunda (VOGA-OX) usa apenas OX.
2. Em VOGA, a taxa de cruzamento foi definida com o valor de 0.8 e taxa de mutação com valor de 0.3. Estes valores foram definidos empiricamente.
3. A função de aptidão adotada foi a função  $g$  do K2 (função (2.3)).
4. Uma vez que os algoritmos tem natureza estocástica, mais que uma execução é necessária para verificar a solução final. Por esta razão, nos resultados apresentados neste trabalho, para cada conjunto de dados, cada algoritmo foi executado 35 vezes. A Tabela 6.9 e a Tabela 6.10 apresentam a pontuação

*Bayesiana* média (função  $g$ ) e o número de gerações necessárias para alcançar a solução, respectivamente.



**Figura 6.8.** Redes *Bayesianas* representando os domínios da Tabela 6.8. As representações gráficas foram criadas usando o software Weka [81].

Tabela 6.9. Pontuações *Bayesianas* (função *g*).

	VOGA	VOEA-RMX	VOEA-OX
<b>Binary-50</b>	-233398	<b>-233395</b>	-233396
<b>Ternary-50</b>	-231050	<b>-231047</b>	-231048
<b>Quaternary-50</b>	-209774	<b>-209769</b>	-209771
<b>Naïve-50</b>	-230709	-230709	-230709
<b>Chain-50</b>	-248406	<b>-248395</b>	-248402
<b>NaiveChain-50</b>	-223755	<b>-223753</b>	-223754
<b>Synth50-50</b>	-1228029	<b>-1227909</b>	-1228047
<b>Synth50-100</b>	-1219929	<b>-1219636</b>	-1220284
<b>Rank (Friedman)</b>	2.625	1.125	2.25

Tabela 6.10. Número de gerações necessárias para convergência.

	VOGA	VOEA-RMX	VOEA-OX
<b>Binary-50</b>	<b>19.08</b>	21.08	20.42
<b>Ternary-50</b>	<b>12.1</b>	12.8	13.18
<b>Quaternary-50</b>	<b>19.37</b>	21.1	20.74
<b>Naïve-50</b>	11.8	<b>11.4</b>	11.58
<b>Chain-50</b>	<b>19.62</b>	23.62	21.37
<b>NaiveChain-50</b>	<b>17.22</b>	21.45	21.74
<b>Synth50-50</b>	<b>15.91</b>	34.42	16.9
<b>Synth50-100</b>	<b>15.45</b>	18.13	16.12
<b>Rank (Friedman)</b>	1.25	2.5	2.25

Considerando os resultados mostrados na Tabela 6.9, é possível tirar algumas conclusões. Levando em consideração todos os domínios avaliados, os valores das pontuações *Bayesianas* (função *g*) obtidos por VOGA-RMX são melhores (ou iguais) do que os alcançados por VOGA. Em geral, para fornecer um quadro melhor do desempenho relativo dos algoritmos sob estudo, foram adicionalmente informados os resultados de comparações estatísticas. Seguindo *Demsar* [76], foi aplicado o bem conhecido teste estatístico de *Friedman* e, quando aplicável, o teste post-hoc de *Nemenyi* para avaliar os resultados obtidos. Sob a hipótese nula, que declara que todos os algoritmos são equivalentes, tem-se  $p = 0.0038$ , assim, a hipótese nula é rejeitada (ou seja, para  $\alpha=5\%$ ). Procedendo com o teste *post-hoc* de *Nemenyi*, verifica-se que o desempenho de VOGA-RMX é significativamente diferente de VOGA nestes conjuntos de dados. É digno mencionar, contudo, que a diferença crítica entre os três *ranks* médios para o teste de *Nemenyi* é igual a 1.17. VOGA-RMX executa significativamente melhor que VOGA ( $2.625-1.125=1.5>1.17$  – veja a última linha da Tabela 6.9) e VOGA-OX não ( $2.625-2.25=0.375<1.17$ ). Assim, é possível concluir que RMX sozinho pode encontrar melhores soluções que a combinação de operadores de cruzamento e mutação

implementada em VOGA, enquanto VOGA-OX não traz resultados significativamente diferentes quando comparados a VOGA.

Outro assunto interessante é o fato de que VOGA-RMX tende a apresentar melhor desempenho (quando comparado a VOGA e VOGA-OX) principalmente em domínios mais complexos (*Synth50-50* e *Synth50-100*).

A Tabela 6.10 revela que VOGA convergiu mais rápido que VOGA-RMX nestes conjuntos de dados. Aplicando o teste estatístico de *Friedman*, tem-se  $p = 0.0302$ , e assim a hipótese nula é rejeitada (para  $\alpha=5\%$ ) e procede-se com o teste de *Nemenyi*. Estes testes mostraram que a convergência de VOGA-RMX é significativamente mais lenta que a de VOGA ( $2.5-1.25=1.25>1.17$  – veja a última linha da Tabela 6.10), enquanto VOGA-OX está apenas abaixo da diferença crítica, mas próximo a ela ( $2.25-1.25=1<1.17$ ). Isto demonstra que RMX não é tão rápido para convergir, mas é capaz de sair de mínimos locais e obter melhores resultados que os outros operadores usados.

#### 6.4.2 Comparação entre RMX e DMO

O operador RMX foi projetado para superar a deficiência do operador DMO quando aplicado a redes *Bayesianas* com estruturas mais esparsas, menos conectadas. Desta forma, RMX pode favorecer todos os tipos de topologias de redes.

Experimentos envolvendo RMX e DMO (aplicando a distância de *Kendall*) foram executados, usando os conjuntos de dados da Tabela 6.8, os quais apresentam estruturas de redes bem diversificadas (veja a Figura 6.8). Os resultados obtidos são mostrados nas Tabelas 6.11 e 6.12 e permitem algumas conclusões.

**Tabela 6.11. Pontuações *Bayesianas* (função g).**

	VOEA-RMX	VOEA-DMO
<b>Binary-50</b>	-233395	<b>-233393</b>
<b>Ternary-50</b>	-231047	<b>-231037</b>
<b>Quaternary-50</b>	<b>-209769</b>	-209774
<b>Naïve-50</b>	-230709	-230709
<b>Chain-50</b>	<b>-248395</b>	-248399
<b>NaiveChain-50</b>	-223753	<b>-223752</b>
<b>Synth50-50</b>	<b>-1227909</b>	-1228135
<b>Synth50-100</b>	<b>-1219636</b>	-1220834

A Tabela 6.11 apresenta os valores das pontuações *Bayesianas* obtidos pelo algoritmo evolutivo (VOEA), usando RMX e DMO isoladamente. É possível notar que VOA-RMX obteve resultados melhores (ou igual) que VOA-DMO em 5 bases de dados. Na base *Quaternary-50*, que possui uma estrutura mais esparsa que *Binary-50* e *Ternary-50*, VOA-RMX mostrou ser melhor que VOA-DMO, assim como era esperado. Já nas bases *Chain-50* e *NaiveChain-50*, nas quais as estruturas apresentam nós conectados em cadeia, os dois operadores comportaram-se de forma semelhante, obtendo resultados bem próximos. Nas bases de dados *Synth50-50* e *Synth50-100*, RMX trouxe melhores resultados que DMO, demonstrando ser mais eficaz em topologias densamente conectadas. Entretanto, ao aplicar o teste estatístico de *Friedman*, tem-se  $p = 0.7$ , e a hipótese nula, que declara que todos os algoritmos são equivalentes, é aceita ( $p > \alpha$ , para  $\alpha=5\%$  ou mesmo para  $\alpha=10\%$ ). Portanto, para estas bases de dados, não foi possível afirmar estatisticamente que RMX é melhor que DMO.

**Tabela 6.12. Número de gerações necessárias para convergência.**

	VOEA-RMX	VOEA-DMO
<b>Binary-50</b>	<b>21.08</b>	29.14
<b>Ternary-50</b>	<b>12.8</b>	24.4
<b>Quaternary-50</b>	<b>21.1</b>	33
<b>Naïve-50</b>	11.4	<b>11.2</b>
<b>Chain-50</b>	<b>23.62</b>	35.45
<b>NaiveChain-50</b>	<b>21.45</b>	22.94
<b>Synth50-50</b>	34.42	<b>13.81</b>
<b>Synth50-100</b>	18.13	<b>11.9</b>
<b>Rank (Friedman)</b>	1.37	1.62

A Tabela 6.12 apresenta o número de gerações obtido por VOA, quando usando RMX e DMO isoladamente. É possível notar que VOA-RMX foi mais eficiente que VOA-DMO, convergindo mais rápido em 5 das 8 bases de dados. Ao ser aplicado o teste estatístico de *Friedman*, obteve-se  $p = 0.0302$ , e a hipótese nula é rejeitada ( $p < \alpha$ , para  $\alpha=5\%$ ). Assim, procedeu-se com o teste de *Nemenyi*. Este teste mostrou que a convergência de VOA-RMX, comparada à de VOA-DMO, está abaixo da diferença crítica ( $1.62-1.37=0.25 < 0.69$  – veja a última linha da Tabela 6.12), portanto, para estes conjuntos de dados, não é possível afirmar estatisticamente que VOA-RMX é mais eficiente que VOA-DMO.

### 6.4.3 Discussão

RMX foi projetado para ser usado como o único operador evolucionário e dinamicamente pode evitar convergência prematura e encontrar boas soluções, em um número razoável de gerações.

Os resultados obtidos nas análises empíricas executadas mostraram que RMX é capaz de melhorar a qualidade das ordenações de variáveis. Além disso, nos conjuntos de dados utilizados, RMX tendeu a encontrar melhores estruturas de redes *Bayesianas* (melhor valor da função  $g$ ), principalmente quando os domínios são grandes e diversificados.

Quando comparado ao operador de mutação DMO, RMX apresentou resultados mais satisfatórios. Contudo, estatisticamente, não houve diferença significativa entre eles com as bases de dados utilizadas nos experimentos.

## 6.5 Algoritmo VOMOS

Recentemente, surgiu um algoritmo adaptativo híbrido capaz de combinar várias abordagens evolucionárias. Este novo método é chamado de *Multiple Offspring Sampling* (MOS) e usa mecanismos oferecidos por heurísticas evolucionárias para criar os descendentes para as próximas gerações, como descrito em [7].

Uma questão importante a ser respondida aqui é se uma forma adequada de integrar diferentes operadores evolucionários (previamente propostos na literatura) pode superar os resultados obtidos por abordagens tradicionais atualmente disponíveis (para a identificação de ordenações de variáveis adequadas ao processo de indução de redes Bayesianas a partir de dados). Portanto, é proposto neste trabalho um MOS, com base na ordenação de variáveis, projetado para melhorar o processo de indução de uma estrutura de rede Bayesiana a partir de dados.

A idéia de usar os operadores embutidos em um algoritmo MOS é explorar o potencial de cada um coletivamente. Neste sentido, o objetivo é abordar o problema de ordenação de variáveis integrando operadores evolucionários definidos anteriormente.

O fato de integrar diferentes métodos e algoritmos para melhorar os resultados em um problema específico tem mostrado bons resultados em trabalhos relacionados a



*Ensembles of Classifiers* [85], assim como no novo paradigma “*Never-Ending Learning*” [86]. Então, surgiu a motivação para explorar estes princípios de integração usando a abordagem evolucionária apresentada em MOS [7].

A idéia principal é explorar o poder de diferentes operadores evolucionários para buscar por ordenações de variáveis adequadas. O novo algoritmo é chamado de VOMOS (*Variable Ordering Multiple Offspring Sampling*) e é apresentado na Figura 6.9 de uma forma geral; a entrada para o algoritmo é o conjunto de dados de treinamento.

VOMOS inicia o processo de busca gerando a população inicial aleatoriamente ( $P_0$ ) e avaliando cada indivíduo. Os indivíduos são avaliados por uma função de aptidão e os melhores são selecionados para a próxima geração. Os novos indivíduos são criados usando um conjunto de operadores de recombinação (cruzamento e/ou mutação). Cada conjunto destes operadores cria seus próprios indivíduos  $O_i^{(j)}$  ( $i$  é a geração e  $j$  é o conjunto de operadores de recombinação).

```

Algoritmo: VOMOS
{Entrada: conjunto de dados de treinamento.}
{Saída: Melhor_VO, Melhor_BN.}
1  begin
2    Crie a população inicial global de soluções candidatas  $P_0$ .
3    Avalie cada população inicial  $P_0$ .
4    while critério de término não é alcançado
5      for para cada operador de cruzamento
6        for para cada operador de mutação (se houver algum)
7          Crie novos indivíduos a partir da população atual  $P_i$ .
8          Avalie cada novo indivíduo.
9          Adicione os novos indivíduos a uma população
            auxiliar  $O_i^{(j)}$ .
10       end
11     end
12     Combine as populações  $O_i^{(j)}$  e  $P_i$  de acordo com um
            critério pré-estabelecido para gerar  $P_{i+1}$ .
13   end
14 end

```

**Figura 6.9. Pseudocódigo do algoritmo VOMOS.**

Neste trabalho, VOMOS foi implementado em duas versões independentes chamadas de VOMOS\_1 e VOMOS\_N, onde os operadores são aplicados a uma população única e a N populações distintas, respectivamente. A Figura 6.10 e a Figura 6.11 apresentam os fluxogramas que detalham estas versões. Em VOMOS\_1, os

operadores de recombinação são combinados e aplicados a mesma população. Em seguida, os melhores indivíduos gerados em cada combinação de operadores são selecionados para criar a próxima geração. Em VOMOS\_N, cada combinação de operadores tem sua própria população. Depois de algumas gerações (por exemplo, cinco gerações), alguns dos melhores indivíduos de cada população são trocados entre todas as populações.

O processo é repetido e, a cada geração, a melhor ordenação é guardada e passada para a próxima. Se não houver melhorias depois de algumas gerações (condição de parada), o algoritmo finaliza e retorna a melhor ordenação encontrada, assim como a rede *Bayesiana* correspondente. Cada indivíduo (isto é, cada ordenação de variáveis) é usado juntamente com o conjunto de treinamento para induzir uma rede *Bayesiana*, a qual é avaliada com base no valor  $g$  (função (2.3)) dado pelo algoritmo K2.

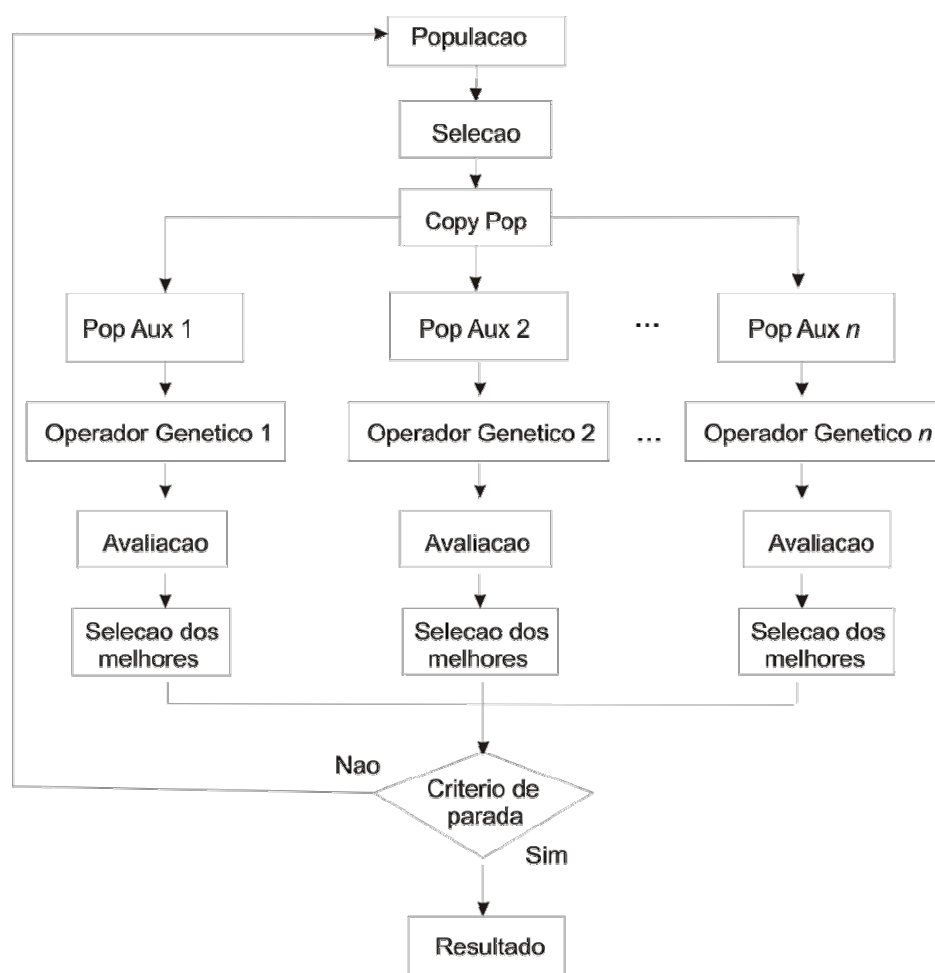


Figura 6.10. Fluxograma do algoritmo VOMOS\_1.

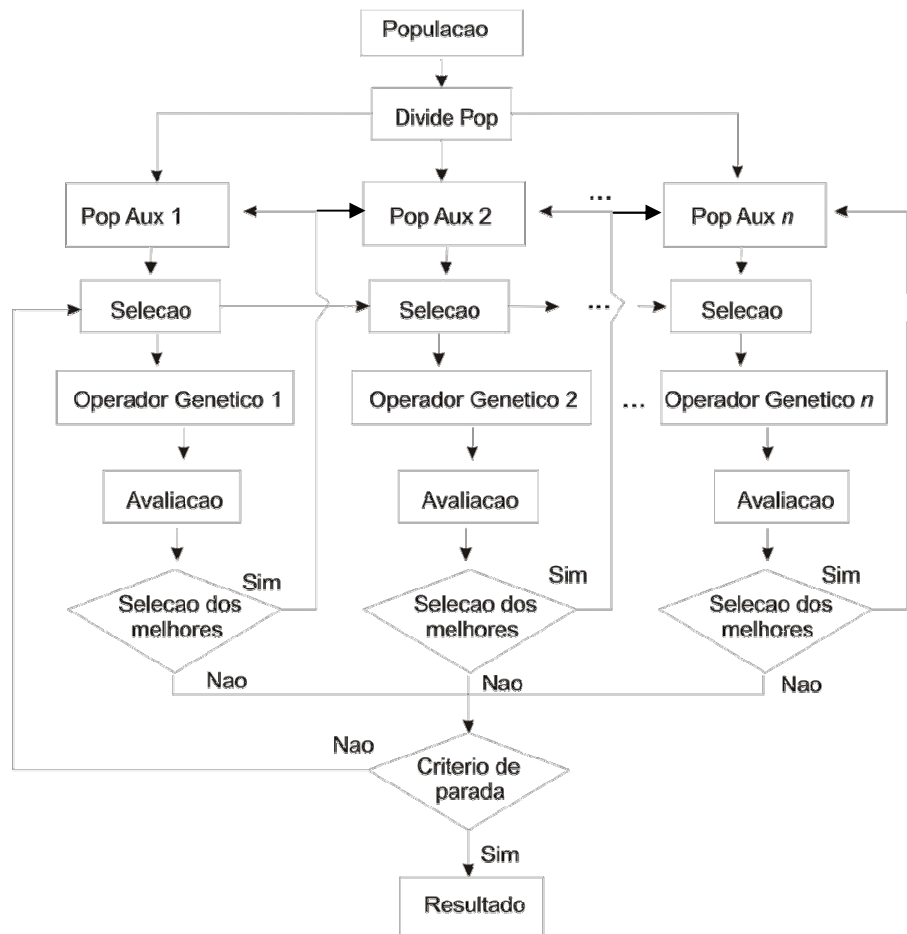


Figura 6.11. Fluxograma do algoritmo VOMOS\_N.

### 6.5.1 Experimentos e Análises de Resultados

Nestes experimentos, foi realizado um estudo sobre o comportamento do algoritmo VOMOS em relação às diferentes combinações de operadores de cruzamento e mutação apresentadas em [47]. Foram escolhidos três operadores de cruzamento – CX, OX2 e POS – e de mutação – DM, ISM e IVM –, os quais obtiveram os melhores resultados nos experimentos dos autores. Em seguida, são comparados o algoritmo evolucionário VOEA-RMX e VOMOS.

### 6.5.1.1 Comparação entre VOMOS e Algoritmos Genéticos

Nos experimentos iniciais, foram desenvolvidos algoritmos genéticos usando diferentes combinações dos operadores genéticos selecionados, como feito em [47], e comparados às versões de VOMOS.

Visando um melhor entendimento da influência dos operadores evolucionários (isto é, cruzamento e mutação), VOMOS deu origem a duas versões independentes, como mencionadas anteriormente. A Tabela 6.13 descreve as diferenças entre estas versões usadas.

Em todos os algoritmos, a taxa de cruzamento usada foi de 0.9 e a taxa de mutação foi de 0.1. Os algoritmos foram executados em duas populações: uma com 10 indivíduos e outra com 50. A função de aptidão aplicada, a qual expressa a qualidade das estruturas, foi a função  $g$  (função (2.3)).

**Tabela 6.13. Principais diferenças entre as duas versões de VOMOS.**

Algoritmo	População inicial	Operadores usados
VOMOS_1	Aleatória e única para todas as combinações de operadores.	Cruzamento: CX, OX2 e POS Mutação: DM, ISM e IVM.
VOMOS_N	Aleatória e replicada para cada combinação de operadores.	Cruzamento: CX, OX2 e POS Mutação: DM, ISM e IVM.

Os experimentos executados usando VOMOS e os algoritmos genéticos envolveram uma simulação da rede *Alarm* [84], consistindo de 5000 casos. Visto que os algoritmos têm natureza estocástica, mais de uma execução é necessária para verificar a solução final. Por esta razão, nos resultados apresentados neste trabalho, cada algoritmo foi executado 35 vezes para o conjunto de dados da *Alarm*.

A média das pontuações *Bayesianas* e o número médio de gerações até a convergência, com as diferentes combinações de operadores genéticos, para os tamanhos de população 10 e 50, são apresentados na Tabela 6.14 e Tabela 6.15, respectivamente. A Tabela 6.16 mostra a média das pontuações *Bayesianas* e o número médio de gerações até a convergência obtidos pelas versões de VOMOS.

Considerando os resultados mostrados nas Tabelas 6.14, 6.15 e 6.16, algumas observações são possíveis. De acordo com o domínio avaliado, os valores das pontuações *Bayesianas* (função  $g$ ) obtidos pelas versões de VOMOS são melhores que os alcançados pelos algoritmos genéticos, combinando diferentes operadores, mesmo em tamanhos de população diferentes. Além disso, é possível notar que quando o

tamanho da população aumenta, as versões de VOMOS melhoram o desempenho médio.

**Tabela 6.14.** Média das pontuações *Bayesianas* (Função g) e número médio de gerações antes da convergência, respectivamente, obtidos pelos algoritmos genéticos com população de tamanho 10, combinando os operadores genéticos.

	<b>CX</b>	<b>OX2</b>	<b>POS</b>
<b>DM</b>	-48665 24	-48636 20	-48620 23
<b>ISM</b>	-48636 31	-48641 27	-48595 28
<b>IVM</b>	-48751 17	-48704 18	-48655 19

**Tabela 6.15.** Média das pontuações *Bayesianas* (Função g) e número médio de gerações antes da convergência, respectivamente, obtidos pelos algoritmos genéticos com população de tamanho 50, combinando os operadores genéticos.

	<b>CX</b>	<b>OX2</b>	<b>POS</b>
<b>DM</b>	-48430 26	-48390 26	-48363 28
<b>ISM</b>	-48466 26	-48320 39	-48356 33
<b>IVM</b>	-48472 23	-48423 26	-48458 25

**Tabela 6.16.** Média das pontuações *Bayesianas* (Função g) e número médio de gerações antes da convergência, respectivamente, obtidos por VOMOS\_1 e VOMOS\_N, com população de tamanho 10 e 50.

<b>Tamanho da população</b>	<b>VOMOS_1</b>	<b>VOMOS_N</b>
<b>10</b>	-48338 26	-48495 18
<b>50</b>	-48097 54	-48302 16

Quando o algoritmo K2 foi utilizado para induzir a estrutura da rede Alarm nestes experimentos, usando uma ordenação considerada ótima, o valor da pontuação *Bayesiana* encontrado foi de -47975. Como pode ser observado nas Tabelas 6.14 e 6.15, nenhuma das melhores ordenações obtidas nas buscas através das diferentes combinações de operadores genéticos foi capaz de superar a avaliação desta ordenação ótima. Contudo, VOMOS\_1 obteve uma avaliação muito próxima do valor obtido quando usando a ordenação ótima (ver Tabela 6.16). Isto indica que a estratégia MOS contribuiu para o alcance de resultados melhores que os obtidos usando as abordagens de algoritmos genéticos tradicionais.

### 6.5.1.2 Comparação entre VOMOS e VOA-RMX

Visto que, nos experimentos anteriores, VOMOS obteve resultados melhores que os algoritmos genéticos tradicionais usando operadores genéticos clássicos, decidiu-se compará-lo com o VOA-RMX, o qual encontrou boas ordenações de variáveis.

Nestes experimentos, os novos indivíduos de VOMOS são criados usando o operador de crossover RMX, apresentado na Subseção 6.4. Este operador mostrou ser promissor e capaz de melhorar a qualidade das ordenações.

Neste novo cenário experimental, VOMOS implementa cinco versões de RMX, de acordo com o número de pontos de corte. Estas versões são: RMX-2 (dois cortes), RMX-3 (três cortes), RMX-4 (quatro cortes), RMX-5 (cinco cortes) e RMX-N ( $n$  cortes). Cada versão de RMX é aplicado a mesma população de VOMOS\_1 e a diferentes populações de VOMOS\_N.

Os experimentos executados usando VOMOS e VOA-RMX envolveram os mesmos domínios usados na Subseção 6.4.1, os quais tem suas características resumidas na Tabela 6.8. Para todos os domínios, com base nas redes *Bayesianas* conhecidas, um conjunto de dados foi gerado usando a técnica *Sampling-from-BN* implementada no *software Weka* [81].

A mesma metodologia experimental usada na Subseção 6.4.1 foi aplicada aqui. A taxa de crossover é de 0.8 e a pontuação *Bayesiana* dada no K2 (função  $g$ ), foi adotado como função de aptidão. Para cada conjunto de dados, cada algoritmo foi executado 35 vezes. A Tabela 6.17 e a Tabela 6.18 apresentam a pontuação *Bayesiana* média e o número médio de gerações necessárias para alcançar a solução, respectivamente.

**Tabela 6.17. Média das pontuações *Bayesianas* (Função  $g$ ).**

	VOEA-RMX	VOMOS_1	VOMOS_N
<b>Binary-50</b>	-233395	-233375	-233385
<b>Ternary-50</b>	-231047	-231016	-231028
<b>Quaternary-50</b>	-209769	-209743	-209752
<b>Naive-50</b>	-230709	-230709	-230709
<b>Chain-50</b>	-248395	-248387	-248395
<b>NaiveChain-50</b>	-223753	-223739	-223748
<b>Synth 50-50</b>	-1227909	-1227813	-1227823
<b>Synth 50-100</b>	-1219636	-1215752	-1216251
<b>Rank (Friedman)</b>	2.8	1.12	2.06

Observando os resultados mostrados na Tabela 6.17, é possível notar que VOMOS\_1 obteve as melhores pontuações *Bayesianas* em todos os domínios avaliados (em *Naive-50* os resultados são iguais para todos os algoritmos). Em geral, para apresentar um quadro melhor dos desempenho relativo dos algoritmos sob estudo, é informado também os resultados comparações de testes estatísticos. Seguindo *Demsar* [76], foi aplicado o conhecido teste estatístico de *Friedman* e, quando aplicável, o teste *post-hoc* de *Nemenyi* para avaliar os resultados obtidos. Sob a hipótese nula, a qual considera que todos os algoritmos são equivalentes, tem-se  $p = 0.005$ , assim a hipótese nula é rejeitada (ou seja, para  $\alpha=5\%$ ) e procede-se com o teste de *Nemenyi*. Estes testes mostram que o desempenho de VOMOS\_1 é significativamente diferente de VOEA-RMX nestes conjuntos de dados. É justo mencionar, contudo, que a diferença crítica entre os três *ranks* médios para o teste de *Nemenyi* é igual a 1.17. VOMOS\_1 executa significativamente melhor que VOEA-RMX ( $2.8-1.12=1.68>1.17$  – veja a última linha da Tabela 6.17) e VOMOS\_N não ( $2.8-2.06=0.74<1.17$ ). Assim, é possível concluir que VOMOS\_1 pode encontrar soluções melhores que VOEA-RMX, enquanto VOMOS\_N não trouxe resultados diferentes significativamente quando comparado a VOEA-RMX.

**Tabela 6.18. Número médio de gerações para a convergência.**

	VOEA-RMX	VOMOS_1	VOMOS_N
<b>Binary-50</b>	21.08	41.73	17.05
<b>Ternary-50</b>	12.8	28.13	16.54
<b>Quaternary-50</b>	21.1	33.92	16.8
<b>Naïve-50</b>	11.4	11.2	11
<b>Chain-50</b>	23.62	46.17	16.6
<b>NaiveChain-50</b>	21.45	47.17	16.51
<b>Synth 50-50</b>	34.42	30.38	16.41
<b>Synth 50-100</b>	18.13	31.76	16.68
<b>Rank (Friedman)</b>	2.12	2.75	1.12

A Tabela 6.18 revela que VOMOS\_N convergiu mais rápido que os outros algoritmos em quase todos conjuntos de dados (pode ter acontecido uma convergência prematura e isto poderia explicar porque esta versão não obteve resultados tão bons quanto os alcançados por VOMOS\_1). Contudo, aplicando o teste estatístico de *Friedman*, tem-se  $p = 0.101$ . Visto que o valor calculado  $p$  é maior que o nível de significância  $\alpha$  (isto é, para  $\alpha=5\%$ ), a hipótese nula não é rejeitada e então os algoritmos são considerados equivalentes com relação à convergência.

### 6.5.2 Discussão

O algoritmo VOMOS foi proposto com base na abordagem MOS e implementado como duas versões independentes chamadas de VOMOS\_1 e VOMOS\_N, as quais aplicam operadores evolucionários a uma única população e a populações diferentes, respectivamente.

Os algoritmos foram avaliados empiricamente usando a base de dados *Alarm* e domínios sintéticos, cujas ordenações ótimas poderiam ser inferidas a partir de suas estruturas originais conhecidas. Resultados iniciais obtidos com VOMOS\_1 e VOMOS\_N foram comparados com resultados obtidos com algoritmos genéticos usando diferentes combinações de operadores genéticos. Para isto, operadores genéticos descritos em [47] como os mais promissores para a busca de ordenações de variáveis foram usados nos experimentos executados. A análise comparativa mostrou a influência do uso destes operadores na taxa de convergência e na qualidade das ordenações obtidas. Em seguida, VOMOS\_1 e VOMOS\_N são comparados a VOA-RMX.

De acordo com os resultados obtidos nos experimentos, VOMOS\_1 supera as outras abordagens com relação à qualidade das ordenações obtidas. Estes resultados demonstram que a combinação de mecanismos para criar a descendência para a nova geração trouxe resultados melhores quando aplicado a uma única população em vez de aplicado a diferentes populações. A respeito da convergência, os testes estatísticos mostraram que VOMOS e as abordagens evolucionárias tradicionais são equivalentes.

## 6.6 Uma Aplicação Prática

Em uma avaliação mais prática, VOMOS foi aplicado a um problema de diagnóstico de processo industrial, juntamente com o algoritmo DMBC, apresentado no Capítulo 5. O desempenho destes métodos foi avaliado sobre os dados de um problema conhecido como *Tennessee Eastman Process* (TEP) [87].

O TEP foi proposto como um teste de controle alternativo e estratégia de otimização para processos químicos contínuos. Este problema é um *benchmark* em engenharia de processos, contendo 52 variáveis, e é apresentado no artigo de *Downs and Vogel* [87] como um problema de controle de uma grande fábrica.



Neste experimento, VOMOS\_1, usando o operador RMX, e DMBC foram aplicados ao TEP com o objetivo de detectar falhas que ocorrem no processo. A idéia era utilizar redes *Bayesianas* para a tarefa de classificação. Embora DMBC fosse mais adequado a este problema, a principal motivação de usar VOMOS\_1 era induzir uma estrutura de classificador *Bayesiano* que representasse melhor as dependências entre as variáveis.

As taxas de classificação obtidas pelos VOMOS\_1 e DMBC são apresentadas na Tabela 6.19. Os resultados são comparados com valores obtidos pelos classificadores *Naive Bayes* e TAN, os quais foram construídos após a aplicação de métodos de seleção de atributos (qui-quadrado –  $\chi^2$  – e *InfoGain*). A aplicação de métodos de seleção de atributos foi sugerida em [88], no qual os autores calcularam a informação mútua entre cada variável do processo e a variável classe.

A Tabela 6.19 mostra que VOMOS\_1 e DMBC obtiveram taxas de classificação superiores às taxas obtidas por *NaiveBayes* e TAN. Como esperado, a estrutura induzida por DMBC (veja a Figura 6.12) foi obtida mais rapidamente que a estrutura aprendida por VOMOS\_1. No entanto, VOMOS\_1 obteve uma estrutura mais elaborada (veja a Figura 6.13), revelando mais relações entre as variáveis. Embora o uso de VOMOS\_1 implique em um esforço computacional mais alto, ele pode ser mais adequado quando o principal interesse é uma identificação mais provável das relações de dependência (ou independência) entre as variáveis.

Tabela 6.19. Taxas de classificação corretas.

Dados	NB (todos os atributos)	TAN (variáveis 9 e 51)	NB (variáveis 9 e 51)	TAN e $\chi^2$	TAN e <i>InfoGain</i>	NB e $\chi^2$	NB e <i>InfoGain</i>	DMBC	VOMOS_1
Treinamento	92.7	88.9	76.8	92.9	93.4	91.7	91.8	92.5	<b>95.4</b>
Teste	73.3	77	77	75	76.5	74.2	76	<b>78.5</b>	76.5

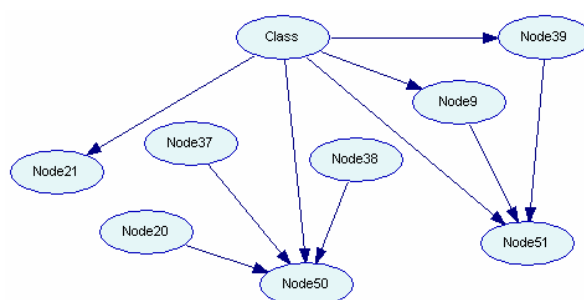


Figura 6.12. Estrutura do classificador DMBC para o TEP.

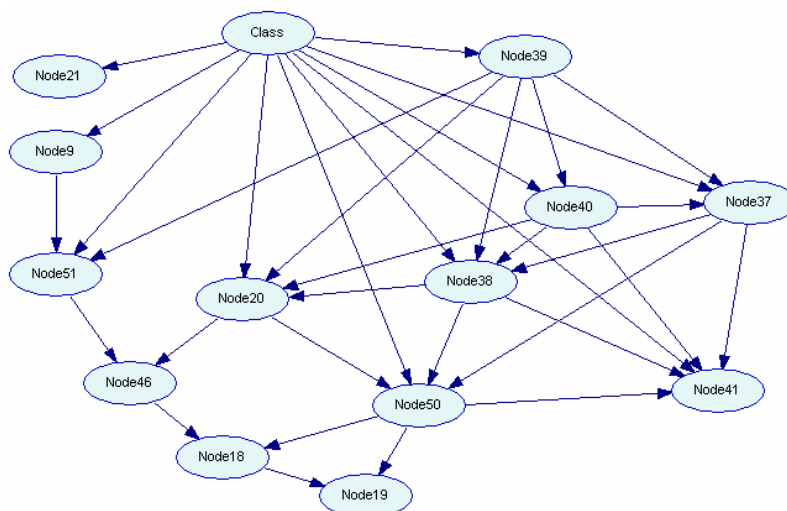


Figura 6.13. Estrutura do classificador VOMOS\_1 para o TEP.

## 6.7 Considerações Finais

O algoritmo VOMOS e os operadores genéticos, DMO e RMX, apresentados neste capítulo, foram desenvolvidos com o objetivo de otimizar o aprendizado de redes *Bayesianas* através da busca de uma ordenação de variáveis adequada. Os resultados obtidos foram promissores para domínios de dados tendo muitas variáveis e sendo densamente conectados. Apenas o operador DMO não se mostrou muito adequado quando a estrutura tende a ser esparsa.

Os algoritmos VOMOS\_1 e DMBC foram ainda aplicados a um *benchmark* conhecido como *Tennessee Eastman Process* (TEP) [87]. O principal objetivo deste experimento era demonstrar o desempenho dos classificadores aprendidos por estes métodos, sem a aplicação de métodos de seleção de atributos. Os experimentos mostraram que VOMOS\_1 e DMBC são capazes de gerar bons classificadores para o TEP e não necessitam da aplicação de métodos de seleção de atributos para escolher as variáveis mais relevantes

## 7 Conclusões e Trabalhos Futuros

O objetivo deste trabalho foi realizar um estudo sobre a ordenação das variáveis no processo de aprendizado de estruturas de redes *Bayesianas* a partir de dados, verificando onde ela é necessária, qual a sua influência e como obtê-la da melhor maneira possível. Consequentemente, métodos de aprendizado de redes *Bayesianas* foram desenvolvidos com base nesta ordenação, visando à otimização deste aprendizado.

Neste estudo, é possível notar que a ordenação das variáveis pode ser considerada uma restrição importante nos algoritmos de aprendizado como forma de reduzir o espaço de busca, embora não seja o ponto mais importante do aprendizado de redes *Bayesianas*.

Como apresentado nas seções anteriores, diversos algoritmos propostos na literatura usam o espaço de ordenações para o aprendizado das redes e não o espaço de estruturas, selecionando para cada ordenação a melhor rede consistente com ela. A maioria destes métodos, entretanto, não são recomendados para domínios contendo muitas variáveis ou para aprender estruturas muito conectadas.

Os métodos desenvolvidos neste trabalho podem ser aplicados a qualquer domínio e podem utilizar ou não a informação da ordenação de variáveis. Se uma rede *Bayesiana* é utilizada apenas para a tarefa de classificação, a ordem ou a direção das variáveis não tem a mesma influência. Contudo, um classificador é apenas uma parte de um processo de decisão maior e, por este motivo, dois novos métodos, chamados DMBC e A-DMBC, foram desenvolvidos neste trabalho com o objetivo de induzir classificadores *Bayesianos* precisos, tendo estimativas de probabilidades de confiança e revelando relações reais entre as variáveis mais relevantes.

Nos casos onde a busca pela melhor ordenação é indicada, esta informação pode ser encontrada com a aplicação de dois novos operadores genéticos desenvolvidos neste trabalho: o operador de mutação DMO e o operador de cruzamento RMX. Além destes dois operadores, um algoritmo evolucionário que aplica a abordagem MOS [7][8] também é proposto.

Os resultados experimentais obtidos por todos os algoritmos desenvolvidos neste trabalho são promissores e demonstram que o objetivo principal foi alcançado.

## 7.1 Avaliação dos Métodos DMBC e A-DMBC

A característica abordada para se otimizar a qualidade das redes *Bayesianas* dentro do processo de aprendizado automático a partir de dados, foi a ordenação das variáveis envolvidas no problema. Entretanto, também foram desenvolvidos neste trabalho dois algoritmos de aprendizado de estrutura que não utilizam esta informação: DMBC e A-DMBC. O objetivo destes algoritmos é induzir classificadores *Bayesianos* a partir de dados.

DMBC e A-DMBC foram projetados com a intenção de relaxar algumas limitações indesejáveis de classificadores *Bayesianos* mais usados, como o *Naive Bayes*, e, ao mesmo tempo, diminuir o tempo computacional necessário para aprender um classificador de rede *Bayesiana*. Isto é feito explorando-se a cobertura de *Markov* (veja SubSeção 2.3) da variável classe, de forma que a indução da estrutura seja mais rápida e produza estimativas mais exatas das probabilidades da classe.

A idéia principal destes algoritmos é reduzir o número de variáveis da rede gerada, diminuindo o número de possíveis estruturas (DAG) a serem investigadas. Assim, o problema da ordenação de variáveis pode ser minimizado. Portanto, DMBC e A-DMBC podem ser vistos como uma alternativa quando não se deseja gastar um tempo elevado na busca pela melhor ordenação.

Comparações com os algoritmos TAN e *Naive Bayes* foram realizadas. As principais conclusões derivadas dos experimentos são: (i) A-DMBC mostrou ser mais eficiente computacionalmente que seus homólogos baseados no K2, com uma redução de cerca de 90% da chamada à função  $g$  (função 2.3); (ii) em muitos conjuntos de dados, as taxas médias de classificação correta (TMCC) são muito próximas às alcançadas pelo TAN, que mostrou alcançar as mais altas TMCC neste estudo; (iii) em muitos conjuntos de dados, o desempenho de A-DMBC na estimação das distribuições de probabilidades é melhor que TAN e *Naive Bayes*, mas é pior que K2. Considerando todas estas razões, é possível acreditar que DMBC e A-DMBC fornecem um bom equilíbrio entre eficiência computacional e precisão na tarefa de induzir classificadores *Bayesianos* a partir de dados.

## 7.2 Avaliação das Estratégias Evolucionárias

Para os casos onde a busca pela melhor ordenação é indicada, este trabalho apresenta novas estratégias evolucionárias com o objetivo de buscar a ordenação adequada para otimizar o aprendizado *Bayesiano*. Inicialmente, foram desenvolvidos dois operadores genéticos específicos para o problema da ordenação: o operador de mutação DMO e o operador de cruzamento RMX. Em seguida, foi criado um algoritmo evolucionário chamado VOMOS, que aplica uma nova abordagem evolucionária conhecida por MOS [7][8].

A idéia do operador DMO é direcionar a busca de acordo com uma distância de permutação entre duas variáveis. Desta forma, DMO vasculha o espaço de busca e procura pela ordenação adequada de uma forma dinâmica. Uma simples permutação de variáveis mais próximas pode levar a busca para uma outra área do espaço de busca, começando em uma nova região, sem mudar a geração drasticamente. Assim, é possível buscar por ordenações adequadas, dinamicamente, evitando ótimos locais.

Os resultados obtidos nas análises empíricas executadas mostraram que DMO é capaz de melhorar a qualidade das ordenações de variáveis quando comparado ao operador de mutação com base em ordem. Além disso, nos conjuntos de dados utilizados, DMO tendeu a encontrar melhores estruturas de redes *Bayesianas* (melhor valor da função  $g$ ), principalmente em domínios maiores e densamente conectados. Contudo, para domínios esparsos, não houve ganhos.

Já o operador de cruzamento RMX não favorece uma topologia de rede específica. RMX escolhe aleatoriamente um número  $n$  de pontos de corte e, dependendo do número de pontos de corte e do tamanho das sequências selecionadas, ele trará vantagens para diferentes topologias.

RMX foi projetado para ser usado como o único operador evolucionário e que pode evitar convergência prematura dinamicamente, encontrando boas soluções, em um número razoável de gerações. Nos conjuntos de dados utilizados, RMX tendeu a encontrar melhores estruturas de redes *Bayesianas*, principalmente quando os domínios eram grandes e diversificados.

O algoritmo VOMOS explora o poder de diferentes operadores de recombinação para gerar os indivíduos da próxima geração. As implementações de VOMOS utilizam

diferentes versões do operador RMX para criar os novos indivíduos e podem aplicá-las à mesma população ou a diferentes populações.

Resultados iniciais obtidos com VOMOS foram comparados a resultados obtidos por algoritmos genéticos usando diferentes combinações de operadores genéticos. A análise comparativa mostrou a influência do uso destes operadores na taxa de convergência e na qualidade das ordenações obtidas.

VOMOS superou as outras abordagens com relação à qualidade das ordenações obtidas. Os resultados demonstraram que a combinação de mecanismos para criar a descendência para a nova geração trouxe resultados melhores quando aplicada a uma única população em vez de aplicada a diferentes populações. A respeito da convergência, os testes estatísticos mostraram que VOMOS e as abordagens evolucionárias tradicionais são equivalentes.

### 7.3 Trabalhos Futuros

Os resultados obtidos com os métodos propostos foram promissores e satisfatórios, no entanto, algumas modificações poderiam trazer novas contribuições.

Testes envolvendo diferentes funções de pontuação (Entropia, BDe (*Bayesian Dirichlet equivalent*), MDL (*Minimum Description Length*) e AIC (*Akaike Information Criterion*)) para serem usados como função de aptidão pelos algoritmos evolucionários foram realizados, contudo não foi encontrada uma função que trouxesse maior ganho que a função  $g$  (função (2.3)), fornecida pelo K2. Uma pesquisa por novas funções, como a proposta em [89], deve ser realizada e novos experimentos, envolvendo os algoritmos evolucionários e estas novas funções, devem ser executados.

Os algoritmos evolucionários desenvolvidos foram implementados com um algoritmo de aprendizado com base em busca e pontuação (algoritmo K2). Uma idéia futura é utilizar também algoritmos de aprendizado com base em independência condicional (como o PC).

## 8 Referências

- [1] PEARL, J. **Probabilistic reasoning in intelligent systems: networks of plausible inference**. San Mateo: Morgan Kaufmann, 1988.
- [2] HRUSCHKA Jr., E. R.; EBECKEN, N. F. F. Towards efficient variables ordering for Bayesian Network Classifiers optimization. *Data & Knowledge Engineering*, v. 63, p. 258-269, 2007.
- [3] CHICKERING, D.M. Learning Bayesian networks is NP-Complete. In: FISHER, D.; LENZ, H (Eds). **Learning from Data: artificial intelligence and statistics V**. Springer-Verlag, p. 121-130, 1996.
- [4] CHICKERING, D.; GEIGER, D.; HECKERMAN, D.E. **Learning Bayesian Networks is NP-Hard**. Research Technical Report MSR-TR-94-17, 1994.
- [5] SPIRITES, P.; GLYMOUR, C.; SCHEINES, R. **Causation, prediction, and search**. New York: Springer-Verlag, 1993.
- [6] PEARL, J. **Causality: models, reasoning, and inference**. Cambridge: University Press, Cambridge, UK, 2000.
- [7] LATORRE, A.; PENA, J. M.; MUELAS, S.; FREITAS, A. A. Learning hybridization strategies in evolutionary algorithms. *Intelligent Data Analysis*, 14: 333-354, 2010.
- [8] LATORRE, A.; PENA, J. M.; GONZÁLEZ, S.; ROBLES, V.; FAMILI, F. Breast cancer bio-marker selection using multiple offspring sampling. In: *Proceedings of the ECML/PKDD. Workshop on Data Mining in Functional Genomics and Proteomics: Current Trends and Future Directions*, Warsaw, Poland, Springer Verlag, 2007.
- [9] HECKERMAN, D. **A tutorial on learning Bayesian networks**. Local: Microsoft Research, Advanced Technology Division, 1995, Notas: Technical Report MSR-TR-95-06.
- [10] BAYES, T. Essay towards solving a problem in the doctrine of chances. **Philosophical Transactions of the Royal Society of London**, v. 53, p. 370-418, 1763.
- [11] NEAPOLITAN, R. E. **Learning Bayesian networks**. New Jersey: Prentice

Hall, Inc., 2003.

- [12] HRUSCHKA, JR.; E. R. **Propagação de evidências em redes Bayesianas:** diagnóstico sobre doenças pulmonares. Março, 1997, 128 p. Dissertação (Mestrado em Ciência da Computação) – Departamento de Ciência da Computação, Universidade de Brasília, Brasília, 1997.
- [13] CASTILLO, E.; GUTIERREZ, J.; HADI, A. **Expert systems and probabilistic network models.** New York: Springer-Verlag, 1996.
- [14] ACID, S.; CAMPOS, L. M. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. **Journal of Artificial Intelligence Research**, v. 18, p. 445-490, 2003.
- [15] CHICKERING, D. M. Optimal structure identification with greedy search. **Journal of Machine Learning Research**, v. 3, p. 507-554, 2002.
- [16] COOPER G.; HERSKOVITZ, E. A Bayesian method for the induction of probabilistic networks from data. **Machine Learning**, v. 9, p. 309-347, 1992.
- [17] DE CAMPOS, L. M.; FERNANDEZ-LUNA, J. M.; PUERTA, J. M. Local search methods for learning bayesian networks using a modified neighborhood in the space of DAGs. In: LECTURE NOTES IN ARTIFICIAL INTELLIGENCE, v. 2527. **Proceedings of the Eight Ibero-American Conference on AI**, p. 182-192, 2002.
- [18] HECKERMAN, D.; GEIGER D.; CHICKERING, D. Learning Bayesian networks: the combination of knowledge and statistical data. **Machine Learning**, v. 20, n. 3, p. 197-243, 1995.
- [19] LAM, W.; BACCHUS, F. Learning Bayesian belief networks: an approach based on the MDL principle. **Computational Intelligence**, v. 10, p. 269-293, 1994.
- [20] RUSSEL, S.; SRINIVAS, S.; AGOGINO, A. Automated construction of sparse Bayesian networks for unstructured probabilistic models and domain information. In: M. HENRION. et al (Eds). **Uncertainty in Artificial Intelligence 5**. North-Holland: p. 295-308, 1990.
- [21] SUZUKI, J. Learning Bayesian belief networks based on the MDL principle: an efficient algorithm using the branch and bound technique. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 1996, Bari.
- [22] COWELL, R. G. Conditions under which conditional independence and scoring



- methods lead to identical selection of Bayesian network models. In: UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, 2001. **Proceedings of the Seventeenth Conference (UAI)**, 2001. San Francisco: Morgan Kaufmann, 2001. p. 91-97.
- [23] DE CAMPOS, L. M. Independence relationships and learning algorithms for singly connected networks. **Journal of Experimental and Theoretical Artificial Intelligence**, v. 10, p. 511-549, 1998.
- [24] DE CAMPOS, L. M.; HUETE, J. F. A new approach for learning belief networks using independence criteria. **International Journal of Approximate Reasoning**, v. 24, p. 11-37, 1998.
- [25] SPIRITES, P.; GLYMOUR, C. An algorithm for fast recovery of sparse causal graphs. **Social Science Computer Review**, v. 9, p. 62-72, 1991.
- [26] VERMA, T.; PEARL, J. Equivalence and synthesis of causal models. In: BONISSONE, P.P. et al (Eds.). **Uncertainty in Artificial Intelligence 6**. North Holland: Elsevier Science Publishers B.V., 1991. p 255-268.
- [27] BOUCKAERT, R. R. Probabilistic network construction using the minimum description length principle. *Lecture Notes in Computer Science*, 747:41–48, 1993.
- [28] COWELL, R. G. et. al. Probabilistic Networks and Expert Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.
- [29] HSU, W. H. Genetic wrappers for feature selection in decision tree induction and variable ordering in Bayesian network structure learning. **Information Sciences**, v. 163, p. 103-122, 2004.
- [30] FRIEDMAN, N. et al. Bayesian network classifiers. **Machine Learning**, v. 29, p. 131-163, 1997.
- [31] FRIEDMAN, N.; KOLLER, D. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks, **Machine Learning**, v. 50, p. 95-125, 2003.
- [32] PENG, F.; SCHUURMANS, D.; WANG, S. Augmenting Naive Bayes Classifiers with Statistical Language Models. *Information Retrieval*, 7, 317–345, 2004.
- [33] SAHAMI, M. Learning limited dependence Bayesian classifiers. In: 2ND INT.

- CONF. KNOWLEDGE DISCOVERY IN DATABASES. **Proc...** Menlo Park, CA: AAAI Press, p. 334-338, 1996.
- [34] GALVÃO, S. D. C. O. ; HRUSCHKA JR., ER . A Markov Blanket based strategy to optimize the induction of Bayesian Classifiers when using Conditional Independence Algoritms. In: THE 9TH INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY (DaWaK 2007), 2007, Rogensburg. Lecture Nontes on Artificial Intelligence. Berlin: Springer, v. 1. p. 1. 2007.
- [35] DUDA, R.; HART, P. **Pattern Classification and Scene Analysis**. New York: Wiley, 1973.
- [36] CHOW, C. K.; LIU, C. N. Approximating discrete probability distributions with dependence trees. **IEEE Transactions on Information Theory**, v. 14, n. 3, p. 462-467, 1968.
- [37] KRUSKAL, J. B. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In: **Proceedings of the American Mathematical Society**, v. 7, n. 1, p. 48–50, 1956.
- [38] PRIM, R. C. Shortest connection networks and some generalizations. In: **Bell System Technical Journal**, v. 36, p. 1389–1401, 1957.
- [39] LARRAÑAGA, P. Probabilistic graphical models and evolutionary computation. 2010 IEEE World Congress on Computational Intelligence, Barcelona (Spain), 2010.
- [40] ROBINSON, R. W. Counting unlabelled acyclic digraphs. In **Lecture notes in Mathematics: Combinatorial Matemathics V**, p. 28-43, Springer Verlag, 1977.
- [41] ETMINANI, K.; NAGHIBZADEH, M.; RAZAVI, A. Globally Optimal Structure Learning of Bayesian Networks from Data. In ICANN'10. **Proceedings of the 20th international conference on Artificial neural networks: Part I**, p.101-106, 2010.
- [42] DALY, R.; SHEN, Q.; AITKEN, S. Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, v. 26, n. 2. p. 99-157, 2011.
- [43] CHICKERING, D.M. Learning equivalence classes of Bayesian networks structures. **The Journal of Machine Learning Research**, v. 2, p. 445-498, 2002.

- [44] ANDERSSON, S. A.; MADIGAN, D.; PERLMAN, M. D. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, v. 25, n. 2, p. 505–541, 1997.
- [45] HRUSCHKA, JR.; E. R.; EBECKEN, N. F. F. Variable Ordering for Bayesian Networks Learning from Data. In: *INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE FOR MODELLING, CONTROL AND AUTOMATION - CIMCA'2003*. Vienna: 2003.
- [46] CHEN, X.; ANANTHA, G.; LIN, X. Improving Bayesian Network Structure Learning with Mutual Information-based Node Ordering in the K2 Algorithm. **IEEE Transactions on Knowledge and Data Engineering**, vol. 20, n 5, p. 628-640, 2008.
- [47] LARRAÑAGA, P. et al. Learning Bayesian network structure by searching for the best ordering with genetic algorithms. **IEEE Trans. on Systems, Man and Cybernetics - Part A: Systems and Humans**, v. 26, n. 4, p. 487-493, 1996.
- [48] WALLACE, C. S.; KORB, K. B. Learning linear causal models by MML sampling. In *Causal Models and Intelligent Data Management*, Gammerman, A. Springer, p. 89-111, 1999.
- [49] ACID, S.; CAMPOS, L. M. A hybrid methodology for learning belief networks: BENEDICT. **International Journal of Approximate Reasoning**, v. 27, n. 3, p. 235-262, 2001.
- [50] SINGH, M.; VALTORTA, M. Construction of Bayesian network structures from data: a brief survey and an efficient algorithm. **International Journal of Approximate Reasoning**, v. 12, n. 2, p. 111–131, 1995.
- [51] DE CAMPOS, L. M.; GÁMEZ, J. A.; PUERTA, J. M. Learning Bayesian networks by ant colony optimization: searching in two different spaces. *Mathware & Soft Computing* v. 9, n. 3, p. 251–268, 2002.
- [52] DE CAMPOS, L. M.; HUETE, J. F. Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing. In: **Proceedings of the Eighth International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00)**, v. 1, p. 333–340, 2000.
- [53] BACK, T. **Evolutionary algorithms in theory and practice**. Oxford University

Press, 1996.

- [54] GRILO, C. **Aplicação de algoritmos evolucionários à extração de padrões musicais**. Novembro, 2002, 192 p. Dissertação (Mestrado em Engenharia Informática) – Departamento de Engenharia Informática, Universidade de Coimbra, Coimbra, 2003.
- [55] HOLLAND, J. H. **Adaptation in natural and artificial systems**. MIT Press, 1975.
- [56] GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning**. Massachusetts: Addison-Wesley, 1989.
- [57] RESENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. Editora Manole.
- [58] LACERDA, E. G. M; DE CARVALHO, A. C. P. L. F. Introdução aos algoritmos genéticos. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 19, **Anais...** Rio de Janeiro. p. 51-126, 1999.
- [59] SYSWERDA, G. Schedule optimization using genetic algorithms. In: DAVIS, L. (Ed). **Handbook of genetic algorithms**. New York: Van Nostrand Reinhold, 1991. p. 332-349.
- [60] DRUZDZEL, M. J. SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: A development environment for graphical decision-theoretic models (Intelligent Systems Demonstration). **Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)**, AAAI Press/The MIT Press, Menlo Park, CA, 1999. p. 902-903.
- [61] PROVOST, F.; FAWCETT, T., Robust classification for imprecise environments. **Machine Learning**, v. 42, p. 203-231, 2001.
- [62] DEGROOT, M.H.; FIENBERG, S.E. The comparison and evaluation of forecasters. *The Statistician* 32, p. 12–22, 1983.
- [63] ZADROZNY, B.; ELKAN, C. Transforming classifier scores into accurate multiclass probability estimates. **Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD'02)**, 2002.
- [64] ROMERO, T.; LARRANAGA, P.; SIERRA, B. Learning Bayesian networks in the space of orderings with estimation of distribution algorithms. **International**

- Journal of Pattern Recognition and Artificial Intelligence**, v. 18, n. 4, p. 607-625, 2004.
- [65] HSU, W. H; GUO, H.; PERRY, B. B. A permutation genetic algorithm for variable ordering in learning Bayesian networks from data. **Proceedings of the 4<sup>th</sup> Annual Conference on Genetic and Evolutionary Computation**, 2002.
- [66] KABLI, R.; HERRMANN F.; McCALL, J. A chain-model genetic algorithm for Bayesian network structure learning. **Proceedings of the 9<sup>th</sup> Annual Conference on Genetic and Evolutionary Computation**, p. 1264-1271, 2007.
- [67] FAULKNER, E. K2GA: Heuristically guided evolution of Bayesian network structures from data. In: COMPUTATIONAL INTELLIGENCE AND DATA MINING (CIDM 2007). IEEE Symposium on March 1, p. 18 – 25, 2007.
- [68] SANTOS, E. B.; HRUSCHKA JR., ER. VOGA: Variable ordering genetic algorithm for learning Bayesian classifiers. In: 6TH INTERNATIONAL CONFERENCE ON HYBRID INTELLIGENT SYSTEMS - HIS2006, 2006, Auckland. **Proceedings of The Sixth International conference on Hybrid Intelligent Systems (HIS06)**. Los Alamitos CA, USA : IEEE Press, 2006.
- [69] SANTOS, E. B.; HRUSCHKA JR., ER; NICOLETTI, M. C. Conditional independence based learning of Bayesian classifiers guided by a variable ordering genetic search. In: THE 2007 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC 2007), 2007, Singapura. **Proceedings of The 2007 IEEE Congress on Evolutionary Computation (CEC 2007)**. Los Alamitos: IEEE Press, v. 1. 2007.
- [70] SANTOS, E. B. **A ordenação das variáveis no processo de otimização de classificadores bayesianos: uma abordagem evolutiva**. 2007, 100 p. Dissertação (Mestrado em Ciência da Computação) – Departamento de Ciência da Computação, Universidade Federal de São Carlos, São Carlos, 2008.
- [71] HRUSCHKA Jr., E. R.; EBECKEN, N. F. F. Missing Values prediction with K2. *Intelligent Data Analysis Journal (IDA)*. Netherlands: v. 6, n. 6, p. 557 - 566, 2002.
- [72] HRUSCHKA Jr., E. R.; GALVÃO, S. D. C. O. Fast Conditional Independence-based Bayesian Classifier. *Journal of Computing Science and Engineering*. v. 1, p.

- 162-176, 2007.
- [73] BESAG, J. Statistical analysis of non-lattice data. *The Statistician*, 24, p. 179–195, 1975.
  - [74] GROSSMAN, D.; DOMINGOS, P. Learning Bayesian Networks Classifiers by Maximizing Conditional Likelihood. **Proceedings of 21st ICML**, Canada, 2004.
  - [75] LANGLEY, P. and SAGE, S. Induction of selective bayesian classifiers. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 1994.
  - [76] DEMŠAR, J. Statistical Comparisons of Classifiers over Multiple Data Sets, *Journal of Machine Learning Research*, p. 1-30, 2006.
  - [77] LARRAÑAGA, P; LOZANO, J. A. Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation, Kluwer Academic, 2001.
  - [78] YANG, S.; CHANG, K.-C. Comparison of score metrics for Bayesian network learning. *Systems, Man and Cybernetics: Part A, IEEE Transactions on* 32 (3), p. 419-428, 2002.
  - [79] BOUCKAERT, R. R. Bayesian Network Classifiers in Weka for Version 3-5-7. [http://www.cs.waikato.ac.nz/~remco/weka\\_bn/](http://www.cs.waikato.ac.nz/~remco/weka_bn/). 2008.
  - [80] AKAIKE, H. Information theory and an extension of the maximum likelihood principle. In B.N. Petrox and F. Caski, editors, *Proceedings of the Second International Symposium on Information Theory*, p. 267-281, Budapest. Akademiai Kiado, 1973.
  - [81] HALL M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, v. 11, n. 3, p. 10-18, 2009.
  - [82] CONOVER, W.J. *Practical Nonparametric Statistics*, third ed. John Wiley & Sons, 1999.
  - [83] MARITZ. J.S. *Distribution-Free Statistical Methods*. Chapman & Hall, p. 217, 1981.
  - [84] BEINLINCH, I. A. *et al.* The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In: *EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE IN MEDICINE*, London.

*Proceedings...* Berlin: Spring Verlag, 1989. p. 247-256, 1989.

- [85] GAMA, J. Combining Classification Algorithms. PhD thesis from University of Porto – Março 2000.
- [86] CARLSON, A.; BETTERIDGE, J.; KISIEL, B.; SETTLES, B.; HRUSCHKA Jr., E. R.; MITCHELL, T. M. Toward an Architecture for Never-Ending Language Learning. Conference on Artificial Intelligence (AAAI), 2010.
- [87] DOWNS, J.; VOGEL, E. Plant-wide industrial process control problem. Computers and Chemical Engineering, v. 17, n. 3, p. 245-255, 1993.
- [88] VERRON, S.; TIPLICA, T.; KOBI, A. Fault diagnosis with bayesian networks: Application to the tennessee eastman process. In IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL TECHNOLOGY (ICIT'06), Mumbai, India. **Proceedings of IEEE International Conference on Industrial Technology (ICIT'06)**. 2006.
- [89] BOUCHAALA, L.; MASMOUDI, A.; GARGOURI, F.; REBAI, A. Improving algorithms for structure learning in Bayesian Networks using a new implicit score. Expert Systems with Applications, v. 37, n. 7, p. 5470-5475, 2010.