Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

A Ordenação das Variáveis no Processo de Otimização de Classificadores Bayesianos: Uma Abordagem Evolutiva

EDIMILSON BATISTA DOS SANTOS

SÃO CARLOS - SP

AGOSTO/2007

A Ordenação das Variáveis no Processo de Otimização de Classificadores Bayesianos: Uma Abordagem Evolutiva

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"A ordenação das variáveis no processo de otimização de classificadores bayesianos: uma abordagem"

EDIMILSON BATISTA DOS SANTOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Estevam Rafael Hruschka Junior
(Orientador – DC/UFSCar)

Profa. Dra. Heloisa de Arruda Camargo
(DC/UFSCar)

Profa. Dra. Maria do Carmo Nicoletti
(DC/UFSCar)

Prof. Dr. Mauricio Fernandes Figueiredo
(DC/UFSCar)

Prof. Dr. Nelson Francisco Favilla Ebecken
(COPPE/UFRJ)

São Carlos
Agosto/2007

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S237ov

Santos, Edimilson Batista dos.
A ordenação das variáveis no processo de otimização de classificadores bayesianos : uma abordagem evolutiva / Edimilson Batista dos Santos. -- São Carlos : UFSCar, 2008.
100 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2007.

1. Redes Bayesianas. 2. Aprendizado do computador. 3. Algoritmos genéticos. I. Título.

CDD: 006.31 (20ª)

Tenho o sonho de que um dia esta nação se erguerá e honrará o verdadeiro significado da sua doutrina: Mantemos ser verdade e evidente que todos os homens são criados iguais.

Martin Luther King

A minha noiva Valquíria, com amor e carinho,
e aos nossos familiares.

Agradecimentos

Agradeço a Deus pela saúde e disposição necessárias para concluir este trabalho.

Ao meu orientador, Prof Dr. Estevam Rafael Hruschka Junior, pela oportunidade de cursar o Mestrado na UFSCar, pela motivação, orientação e entusiasmo, fazendo-me trabalhar com dedicação e prazer, mesmo diante das dificuldades.

Aos meus pais, Geraldo e Maria dos Anjos, e irmãos, Roberto e Micaela, que sempre me apoiaram.

A minha namorada e noiva, Valquíria, que me incentivou desde o início e suportou a minha ausência, com paciência e compreensão, mesmo em momentos difíceis. E a sua família, que sempre tiveram o maior carinho comigo.

Aos meus amigos, que conviveram comigo nas repúblicas que morei (César Akira, Diego, Eric, Leandro, Luizão, Ricardo, Roberto Rigolin, Rodolfo Carapelli, Rodolfo Golombieski, Rodrigo, Tiago, Tiagão, entre outros), nas salas de aula, no laboratório (Sebastian), pelas contribuições e pela companhia.

Aos meus amigos da empresa Cientistas Associados, pelas oportunidades e pelos bons momentos de companheirismo.

Aos professores da minha cidade, com carinho especial a D. Dola, que me ensinaram a sonhar com a universidade e me incentivaram a conquistar uma vaga.

Aos professores do Departamento de Computação da Universidade Federal de Lavras, que se preocupam em formar bons profissionais e me mostraram o caminho da pesquisa.

Aos professores do Departamento de Computação da UFSCar, principalmente aos que tive a oportunidade de conhecer durante estes anos de estudo.

A todos os funcionários, pelos serviços prestados e todo auxílio concedido.

RESUMO

Classificação é uma tarefa importante em análises de dados e reconhecimento de padrões e requer a construção de um classificador. A indução de classificadores a partir de um conjunto de dados é um problema importante em aprendizado de máquina. Diversas abordagens para a resolução deste problema se baseiam em várias representações, como árvores de decisão, redes neurais, grafos de decisão e regras. Entretanto, têm crescido bastante o interesse em métodos *Bayesianos* para classificação. Os algoritmos de aprendizado de redes *Bayesianas* podem ser usados para induzir classificadores *Bayesianos*. Contudo, o aprendizado de redes *Bayesianas* a partir de dados é um problema NP-Completo e não há métodos computacionais capazes de identificar a melhor solução para todos os problemas de aplicação. Uma restrição comum nestes algoritmos de aprendizado é a ordenação prévia das variáveis utilizadas na definição do problema. As ordenações das variáveis representam os possíveis relacionamentos entre as variáveis na formação da estrutura da rede *Bayesiana* que descreve o problema. Utilizando uma ordenação adequada das variáveis, os algoritmos de aprendizado são capazes de encontrar uma solução mais eficiente. Sendo assim, são propostos, neste trabalho, métodos híbridos para identificar uma ordenação adequada de variáveis, visando à otimização do aprendizado de redes *Bayesianas* para a tarefa de classificação. Os métodos propostos, chamados de VOGA, VOGAC e VOEA, utilizam algoritmos evolucionários e algoritmos de aprendizado de redes *Bayesianas*. Estes métodos usam a informação da variável classe na definição da ordenação mais adequada. Os experimentos executados em alguns domínios de bases de dados revelaram que os métodos propostos são promissores.

ABSTRACT

Classification is a basic task in data analysis and pattern recognition that requires the construction of a classifier. The induction of classifiers from data sets is an important problem in machine learning. Numerous approaches to this problem are based on various representations such as decision trees, neural networks, decision graphs, and rules. However the interest in Bayesianos methods for classification has grown sufficiently. Bayesian Networks (BNs) learning algorithms can be used to induce Bayesian classifiers. However, BNs learning from data is known to be a NP problem and does not have computational methods capable to identify to the best solution for all the application problems. A very common restriction when learning a BN is the definition of a previous Variables Ordering (OV). The OV represent the possible relationships between the variables in the formation of the structure of BN that describes the problem. Using an adequate OV, learning algorithms are capable to find a solution more efficient. Therefore, this work proposes hybrid approaches to help the process of learning a BN from data for classification. The proposed methods named VOGA, VOGAC e VOEA uses Evolutionary Algorithms to optimize the BN learning process by means of the identification of an adequate variables ordering. These methods use information about the class variable when defining the most suitable variable ordering. Experiments performed in a number of datasets revealed that methods are promising.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A classificação é uma tarefa importante no processo de mineração de dados. Consiste em atribuir uma classe a instâncias descritas por um conjunto de atributos. Para isto, é necessária a construção de um classificador, a partir de um conjunto de dados de instâncias pré-classificadas, o que é um problema central em aprendizado de máquina [20].

Diversas abordagens para este problema têm sido baseadas em vários modelos, como árvores de decisão, redes neurais, grafos de decisão e regras. Além destes modelos, o uso de métodos *Bayesianos* para classificação de têm crescido bastante[54].

Redes *Bayesianas* [49] são considerados meios populares de representar graficamente as dependências que existem em um domínio. Estes modelos ajudam a identificar dependências entre variáveis (ou atributos) de uma base de dados, sendo úteis para tarefas de classificação. O classificador *Bayesiano Naive* [22] é um bom exemplo, na literatura, de uma técnica de classificação simples e poderosa.

Os algoritmos de aprendizado de redes *Bayesianas* podem ser usados para induzir classificadores *Bayesianos*. Em vez de codificar uma distribuição de probabilidade conjunta sobre um conjunto de variáveis aleatórias, um classificador *Bayesiano* tem como objetivo predizer corretamente o valor de uma variável classe discreta, dado um vetor de atributos.

Embora o aprendizado de redes *Bayesianas*, assim como o classificador *Bayesiano Naive*, tenham sucesso em diferentes domínios, eles têm suas desvantagens [54]. O classificador *Naive* é muito eficiente para inferência, mas faz fortes suposições de independência entre variáveis/atributos, que muitas vezes não acontece na prática e pode levar a uma estimativa de probabilidade pouco confiável. Já o aprendizado de redes *Bayesianas* frequentemente consome muito tempo e torna-se intratável à medida que o número de variáveis no domínio cresce.

O aprendizado de redes *Bayesianas* a partir de dados se tornou uma área de pesquisa bastante efervescente nos últimos anos [30]. Encontrar a estrutura de rede que melhor represente as dependências entre as variáveis é uma tarefa difícil, pois o espaço de busca para uma rede com *n* variáveis tem dimensão exponencial. Por ser um problema NP-Completo [12][13], não há métodos computacionais capazes de identificar a melhor solução para todos os problemas de aplicação.

Buscando-se reduzir o espaço de busca deste processo de aprendizado, algumas restrições são impostas e, assim, os algoritmos conseguem 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 [61]. Assim, a estrutura gráfica de uma rede *Bayesiana* passa a depender da ordenação das variáveis, o que faz com que a estrutura gráfica varie na dependência da ordenação adotada.

Uma forma de realizar a busca da ordenação de variáveis é por meio de técnicas de computação evolutiva. Na literatura, são encontrados alguns trabalhos que descrevem modelos computacionais evolutivos para encontrar uma ordenação de variáveis adequada para o aprendizado de redes *Bayesianas* [35][31][52]. Neste trabalho, é proposto o desenvolvimento de modelos computacionais evolutivos que buscam uma ordenação adequada de variáveis, visando otimizar o aprendizado de redes *Bayesianas* para serem utilizadas na tarefa de classificação.

1.1 Justificativa e Objetivos

O aprendizado de redes *Bayesianas* não é uma tarefa trivial. Muitos esforços têm sido dedicados ao desenvolvimento de métodos para resolvê-la. 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.

Uma vez que não existem métodos automáticos computacionalmente eficientes na busca de uma ordenação de variáveis adequada, o objetivo técnico deste trabalho foi propor, implementar e avaliar métodos para a otimização do aprendizado supervisionado de redes *Bayesianas* a partir de dados para a classificação, por meio da identificação de uma ordenação adequada das variáveis envolvidas na definição do problema. Para tanto, utilizou-se técnicas de computação evolutiva.

1.2 Metodologia

Neste trabalho, foram utilizadas técnicas de computação evolutiva na busca de uma ordenação adequada de variáveis para a otimização do aprendizado de redes *Bayesianas,* utilizadas na tarefa de classificação.

A computação evolutiva busca a criação de sistemas para a resolução de problemas que utilizam modelos computacionais baseados na teoria da evolução natural e na genética. Existem vários modelos computacionais evolucionários que foram propostos e estudados, e aos quais se dá a denominação de algoritmos evolucionários.

Como proposto em [2], a computação evolutiva pode ser dividida em 3 principais categorias: Algoritmos Genéticos, Estratégias de Evolução e Programação Genética. Os algoritmos genéticos constituem uma das mais conhecidas classes de algoritmos

evolucionários, sendo 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 ferramenta de busca e otimização para a solução dos mais diferentes tipos de problemas [21].

As características dos algoritmos evolucionários os tornam adequados na busca de uma boa ordenação de variáveis. Neste trabalho, os algoritmos evolucionários desenvolvidos geram indivíduos que representam as ordenações possíveis e as avaliam com a ajuda de algoritmos de aprendizado de redes *Bayesianas*. Estes métodos híbridos foram chamados de: VOGA (*Variable Ordering Genetic Algorithm*) [55], VOGAC (*Variable Ordering Genetic Algorithm Classification*) [56] e VOEA (*Variable Ordering Evolutionary Algorithm*).

1.3 Organização do Trabalho

O Capítulo 1, deste trabalho, apresenta uma introdução sobre o uso de redes *Bayesianas* na tarefa de classificação, enfatizando o emprego de uma ordenação prévia das variáveis do problema, visando à otimização dos algoritmos de aprendizado das redes.

O Capítulo 2 destaca os conceitos fundamentais da teoria de redes *Bayesianas*, incluindo a definição de inferência e a descrição do processo de aprendizado. Como exemplos, são apresentados alguns algoritmos de aprendizado de redes *Bayesianas* e de classificadores. Além disso, descreve o problema e a importância da ordenação das variáveis do problema.

O Capítulo 3 apresenta uma introdução sobre computação evolutiva e as características dos algoritmos evolucionários. É destacado o conceito de algoritmos genéticos, apresentando a forma de representação do problema e os operadores de cruzamento e mutação. Um exemplo didático ilustra sua aplicação para melhor compreensão.

No Capítulo 4, são apresentadas algumas abordagens, encontradas na literatura, que utilizam algoritmos evolucionários no aprendizado de redes *Bayesianas*. Alguns métodos são híbridos e exploram domínios específicos da aplicação de redes *Bayesianas*. Já outros, são voltados para o aprendizado de redes *Bayesianas* irrestritas. Também tem métodos que se dedicam ao problema da busca de ordenação de variáveis para otimização do aprendizado.

No Capítulo 5, são detalhados aspectos dos métodos híbridos propostos, os métodos VOGA, VOGAC e VOEA, e os resultados obtidos nos experimentos realizados. Especificamente, são descritas as bases de dados usadas, os resultados obtidos e discussões. Também, é apresentado um novo algoritmo de aprendizado de classificadores *Bayesianos*, denominado K2-MBC.

E, por fim, o Capítulo 6 encerra a descrição do trabalho de pesquisa desenvolvida, apresentando as conclusões e destacando-se as contribuições, as perspectivas atuais e as sugestões de trabalhos futuros.

2 REDES BAYESIANAS

Formalmente, as redes *Bayesianas* são grafos direcionados acíclicos (estrutura), nos quais os nós representam variáveis aleatórias com medidas de incerteza associadas (parâmetros numéricos). Os arcos representam a existência de uma influência direta entre as variáveis conectadas, e a "força" destas influências é quantificada por probabilidades condicionais [49].

Uma rede *Bayesiana* constitui um modelo de processo de raciocínio, uma forma de representação de conhecimento, assim como sistemas baseados em regras e redes neurais, por exemplo. Redes *Bayesianas* são representações gráficas de distribuição de probabilidades e utilizam o Teorema de *Bayes* [3] como método quantitativo para a revisão dessas probabilidades, com base em uma nova informação amostral.

Nas últimas décadas, essas representações têm sido muito utilizadas para representação de incerteza em inteligência artificial. Segundo [25], elas representam um papel crucial em modernos sistemas especialistas, máquinas de diagnósticos e sistemas de suporte à decisão.

A motivação para utilização de redes *Bayesianas* está no fato dos seres humanos terem certa necessidade de moldar os fatos e fenômenos em forma de relacionamentos causais. Sempre que um fato está sendo analisado, busca-se uma explicação que seja a causa do fato em questão [24].

A estrutura da rede *Bayesiana* da Figura 2.1 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 [49] 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).

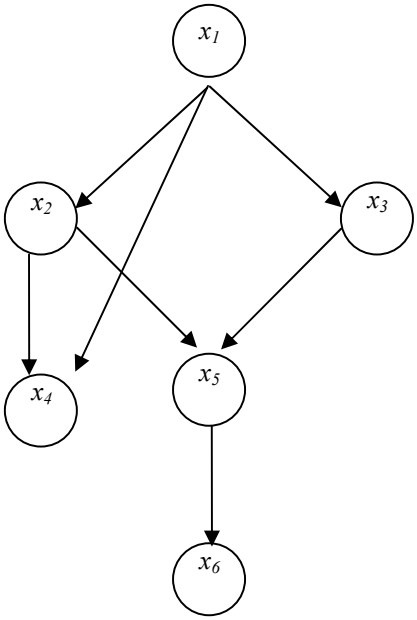


Figura 2.1. Estrutura de Rede *Bayesiana*.

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(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 (1):

$$P(x_1, x_2, ..., x_6) = P(x_6/x_5)\, P(x_5/x_2, x_3)\, P(x_4/x_1, x_2)\, P(x_3/x_1)\, P(x_2/x_1)\, P(x_1) = \prod_{i=1}^{m} P(x_i / \pi_{x_i}), \qquad (1)$$

na qual m é o número de variáveis, $P(x_1, x_2, ..., x_6)$ é a probabilidade conjunta de $X = \{ x_1, x_2, ..., x_6\}$, 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 π_{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 [25].

Partindo-se do conceito de independência, pode-se definir um conjunto de nós da rede *Bayesiana* que têm influência sobre uma determinada variável do problema. Para tanto, considere uma rede *Bayesiana* onde Λ_A é o conjunto de filhos do nó (variável) A e π_A é o conjunto de pais da variável A. O conjunto de nós formado pela união dos conjuntos Λ_A, π_A e os pais das variáveis contidas em Λ_A é chamado de *Markov Blanket* da variável A. Um exemplo genérico pode ser visto na Figura 2.2.

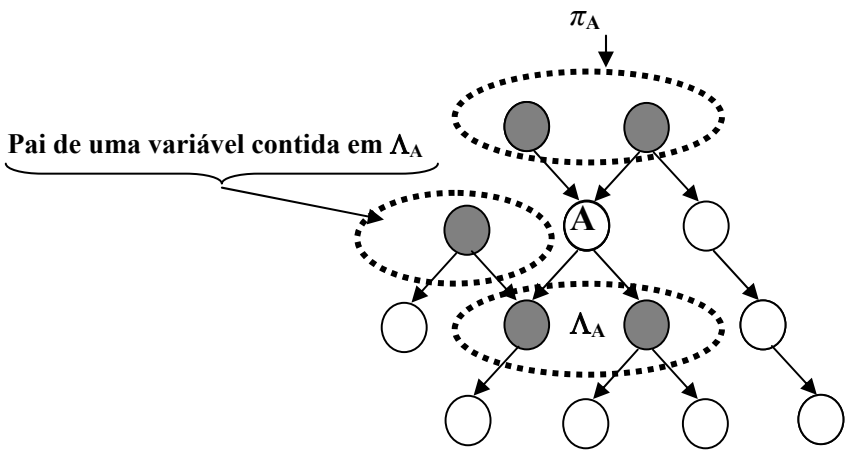


Figura 2.2. Os nós hachurados representam o *Markov Blanket* do nó A.

Como pode ser visto em [49], 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 no *Markov Blanket* de A.

Uma vez que a rede *Bayesiana* esteja definida, inferências podem ser realizadas acerca do conhecimento representado pela estrutura. A Seção seguinte aborda alguns conceitos de inferências.

2.1 Inferência

Tendo-se uma rede *Bayesiana* definida, pode-se extrair os conhecimentos nela

representados através de um processo chamado inferência. De uma maneira geral, a inferência em redes *Bayesianas* 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 os fatos necessários (evidência), propagar estes fatos por toda a rede e, em seguida, consultar o estado da variável alvo. A propagação dos fatos (ou propagação de evidências) é responsável pela atualização do conhecimento a partir de informações fornecidas à rede.

Existem vários métodos para a realização da propagação de evidências em uma rede *Bayesiana* [29]. Dos métodos tradicionais destacam-se: i) propagação em poliárvores, que são redes *Bayesianas* onde um nó pode ter múltiplos pais, como na Figura 2.3, contudo não existe mais que um caminho entre dois nós. Estas redes são chamadas poliárvores porque podem ser vistas como uma coleção de várias árvores que se fundem nos nós onde as setas convergem [61]; ii) eliminação de variáveis [17] e iii) variações de ambos. Como o objetivo deste projeto está vinculado com o aprendizado de redes *Bayesianas*, os métodos de inferência não serão abordados com maiores detalhes. Os leitores mais interessados neste assunto podem consultar [29] e [45].

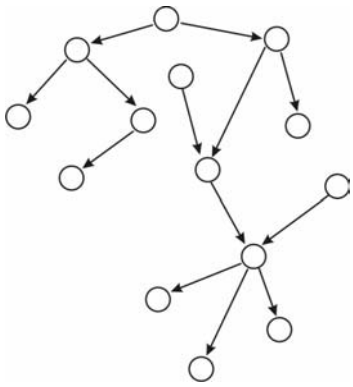


Figura 2.3. Um fragmento de uma poliárvore.

2.2 Aprendizado de Redes *Bayesianas*

O aprendizado de redes de conhecimento[1] pode ser visto como um processo que gera uma representação interna (na máquina) das restrições que definem um dado problema, de modo a facilitar a recuperação dos dados, gerando assim um menor esforço computacional para este tipo de representação [49].

Para modelar uma rede *Bayesiana* que representa um problema, muitas dificuldades são encontradas, entre elas, o processo de entrevistar o especialista para extrair conhecimento especializado. A obtenção de informações claras e precisas da maneira como os especialistas solucionam problemas não é uma tarefa trivial, pois o ser humano demonstra uma certa dificuldade em associar números (dados estatísticos) ao quanto crê em um determinado fato [48]. Além disso, nem sempre o engenheiro do conhecimento conhece a área de aplicação (domínio) em questão, o que dificulta muito a aquisição de informações. Por isso, processos de geração automática de redes *Bayesianas* podem ser implementados para a aquisição do conhecimento a partir de dados.

O processo de aprendizado em redes *Bayesianas* deve: identificar a sua estrutura (B_s), ou seja, identificar as relações de interdependência dadas pelos arcos e induzir as distribuições de probabilidades (B_p) da rede. Normalmente, este processo é dividido em dois subprocessos: "aprendizado da estrutura" e "aprendizado dos parâmetros numéricos" [8].

O aprendizado da estrutura, B_s, é a identificação das dependências e independências das variáveis e da direção da causalidade. O aprendizado dos parâmetros numéricos, B_p, ou das probabilidades, é a identificação da "força" dos relacionamentos, através da probabilidade condicional para cada nó, dados a estrutura B_s e os dados.

---

[1] As redes *Bayesianas* podem ser vistas como um tipo específico de rede de conhecimento, neste texto, entretanto, *redes Bayesianas* e redes de conhecimento são tratadas como sinônimos.

O primeiro resultado encontrado na literatura com uma estrutura de aprendizado de redes de conhecimento a partir de dados[2], foi através do método de *Chow* e *Liu* [14], que trabalha com estrutura de árvore para *k* variáveis. Este método estima uma distribuição de probabilidade conjunta *P* e procura a árvore de conhecimento geradora de uma distribuição de probabilidade *P'* que seja a mais próxima de *P*.

Rebane e *Pearl* [50] criaram um algoritmo, para ser utilizado juntamente com o método de *Chow* e *Liu*, chamado "algoritmo de recuperação de poliárvores" destinado aos casos em que a representação gráfica da distribuição *P* é dada em forma de uma poliárvore. Então, através do método de *Chow* e *Liu*, é gerada a estrutura básica da árvore e em seguida, aplicando-se a esta estrutura o algoritmo de recuperação de poliárvores, obtém-se a representação gráfica da distribuição. Este algoritmo só recupera a poliárvore caso ela seja um mapeamento perfeito da distribuição a ser representada [49].

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 [1], [5], [11], [15], [26], [34], [53] e [62]. Já na segunda classe, estão os algoritmos que utilizam o conceito de independência condicional [49] para a construção da rede; trabalhos que aplicam esta classe de algoritmos, podem ser encontrados em [61], [4], [6], [16], [60] e [65]. 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 [62], [4], [6] e [16].

Os algoritmos de busca heurística vêem o problema de aprendizado como um problema de busca pela estrutura que melhor representem os dados. Para buscá-la, há dois tipos de algoritmos de busca: *forward* (inicia com um grafo desconectado e vai adicionando,

[2] Para efeito de simplicidade, no restante deste texto, o aprendizado de *redes Bayesianas* a partir de dados será chamado aprendizado de *redes Bayesianas*.

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). Em seguida, os algoritmos de busca utilizam um método de pontuação para identificar se a nova estrutura é melhor do que a antiga. Este processo se repete até que a melhor estrutura seja encontrada.

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 (χ^2) [45], utilizando uma base de dados que representa o domínio.

A seguir, são apresentados dois algoritmos que caracterizam cada uma das duas classes principais de algoritmos de aprendizado de estruturas: o primeiro é o algoritmo K2, pertencente à categoria de busca heurística, e o segundo é o algoritmo PC, pertencente à categoria dos algoritmos que utilizam o conceito de independência condicional.

2.2.1 Algoritmo K2

K2 é um algoritmo de busca heurística, proposto por *Cooper* e *Herskovitz* [15], para o aprendizado de estruturas de redes *Bayesianas*, tendo, como entrada, um conjunto de dados e uma ordenação das variáveis e gera, como saída, a estrutura da rede *Bayesiana*.

O algoritmo K2 aplica um método de pontuação *Bayesiano* para cada estrutura testada e seu objetivo é encontrar a estrutura de rede *Bayesiana* mais provável no espaço de busca, dado um conjunto de dados. 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 [31][65].

Uma estrutura de rede *Bayesiana,* B_s, é acompanhada por probabilidades condicionais (parâmetros numéricos), B_p, para formar uma rede *Bayesiana B.* Assim, $B = (B_s, B_p)$. Para cada nó[3] na estrutura, há uma função de probabilidade condicional, se um nó é predecessor imediato (pai) de outro. Se um nó não tem pai, então uma função de probabilidade a *priori, P(x),* é especificada. Um exemplo de um possível conjunto de probabilidades é mostrado na Tabela 2.1 para a estrutura da rede da Figura 2.4 (adaptada de [15]), onde x_1, x_2 e x_3 são variáveis aleatórias que podem assumir valores 1 e 0. A estrutura B_{s_1}, na Figura 2.4, e as probabilidades B_{p_1}, na Tabela 2.1, definem uma rede *Bayesiana*, denotada como B_1.

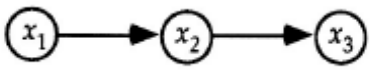


Figura 2.4. Exemplo de rede *Bayesiana* com 3 variáveis, denotada como B_{s_1}.

Tabela 2.1. Atribuições de probabilidade associadas com a estrutura da rede B_{s_1} na Figura 2.4. Essas probabilidades foram chamadas de B_{p_1} (adaptada de[15]).

$P(x_1 = 1) = 0,6$	$P(x_1 = 0) = 0,4$
$P(x_2 = 1 / x_1 = 1) = 0,8$	$P(x_2 = 0 / x_1 = 1) = 0,2$
$P(x_2 = 1 / x_1 = 0) = 0,3$	$P(x_2 = 0 / x_1 = 0) = 0,7$
$P(x_3 = 1 / x_2 = 1) = 0,9$	$P(x_3 = 0 / x_2 = 1) = 0,1$
$P(x_3 = 1 / x_2 = 0) = 0,15$	$P(x_3 = 0 / x_2 = 0) = 0,85$

A Figura 2.4 mostra um exemplo de rede *Bayesiana* contendo 3 variáveis. A Tabela 2.2 mostra uma base de dados fictícia usada para o aprendizado desta rede. Um arco de x_1 para x_2 indica que essas variáveis são probabilisticamente dependentes. A ausência de um arco de x_1 para x_3 indica que não há dependência probabilística direta entre essas duas

[3] A palavra "nó" refere-se aos nós de uma rede *Bayesiana*; a palavra "variável" refere-se às variáveis de um problema; e a palavra "atributo" refere-se aos atributos de uma base de dados. Entretanto, no processo de aprendizado de uma rede *Bayesiana*, estes termos referem-se ao mesmo conceito e por isso podem ser utilizados como sinônimos.

variáveis. A representação de dependências e independências é a função essencial da rede *Bayesiana* [15].

Tabela 2.2. Um exemplo de base de dados, adaptada de[15].

	Valores das variáveis para cada caso		
Caso	x_1	x_2	x_3
1	1	0	0
2	1	1	1
3	0	0	1
4	1	1	1
5	0	0	0
6	0	1	1
7	1	1	1
8	0	0	0
9	1	1	1
10	0	0	0

A estrutura mais provável encontrada maximiza $P(B_s / D)$ (probabilidade de B_s, de acordo com uma base de dados D). Ao tentar maximizar $P(B_s / D)$, sobre todas as possíveis B_s, para a base de dados da Tabela 2.2, é encontrada a estrutura $x_3 \rightarrow x_2 \rightarrow x_1$ como a mais provável. Mais detalhes sobre como maximizar $P(B_s / D)$ podem ser encontrados no Exemplo 1, no final desta subseção.

A dificuldade em se determinar qual a melhor estrutura B_s, que maximiza $P(B_s / D)$, está no número de estruturas possíveis que cresce exponencialmente de acordo com o número de nós. Assim, uma busca exaustiva de todas as estruturas de redes não é computacionalmente viável em muitos domínios.

Em [15], foi proposto um método de busca heurística para maximizar $P(B_s, D)$. Suponha x_i uma variável da rede. Os pais de x_i são representados por uma lista (vetor) de variáveis, denotada por π_i. w_{ij} é usado para designar a *j-ésima* instanciação única de valores das variáveis em π_i, relativa a alguma ordenação de casos na base de dados D, que pode ser a ordem da base, não sendo a única possível. É dito que w_{ij} é um valor ou uma instanciação de π_i. Por exemplo, considere o nó x_2 em B_{s_1}, tendo a Tabela 2.2 como base de dados. O nó x_1 é o pai de x_2 em B_{s_1} e assim $\pi_2 = \{x_1\}$. Nesse exemplo, $w_{21} = 1$, porque, na Tabela 2.2, o primeiro

valor de x_1 é o valor *1*. Além disso, $w_{22} = 0$, porque o segundo valor de x_1, na Tabela 2.2, (relativo à ordenação dos casos na tabela) é o valor *0* (a variável x_1 só pode ter dois valores nesse exemplo: *1*e *0*).

Suponha ainda que *Z* seja um conjunto de *n* variáveis discretas, na qual uma variável x_i em *Z* tem r_i valores possíveis: ($vi_1, vi_2, ..., vi_{ri}$). Considere *D* uma base de dados com *m* casos, onde cada caso contém um valor para cada variável em *Z*. Suponha também que B_s denote uma estrutura de rede *Bayesiana*, contendo apenas as variáveis em *Z*. Suponha que há q_i instanciações de π_i. Pode-se definir N_{ijk} como o número de casos em *D*, onde cada variável x_i tem o valor v_{ik} e π_i é instanciado como w_{ij}. Assim:

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \tag{2}$$

De acordo com as suposições dadas em [15], segue que:

$$P(B_s, D) = P(B_s) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{3}$$

Suponha que há uma ordenação sobre os domínios das variáveis, tal que, se x_i precede x_j na ordenação, então não são permitidas estruturas nas quais exista um arco de x_j para x_i. (veja detalhes da ordenação de variáveis na Seção 2.4). Dada tal ordenação como restrição, haverá $2^{\binom{n}{2}} = 2^{n(n-1)/2}$ estruturas de redes possíveis e não é possível aplicar a equação (3) para cada uma destas estruturas.

Considerando, além da ordenação, que, *a priori*, todas as estruturas são consideradas igualmente prováveis, o algoritmo K2 utiliza um método de busca *greedy* para maximizar $P(B_s / D)$, assim, a partir de (3), é definida a função *g* dada por (4):

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{4}$$

na qual N_{ijk} (2) é calculado de acordo com a base de dados *D* e com π_i, que são os pais de x_i.

K2 começa a busca da estrutura mais provável, tornando a primeira variável da ordenação das variáveis de entrada o nó inicial. Inicialmente, é suposto que um nó não tem

pais e, incrementalmente, são adicionados nós pais a ele, de acordo com a ordenação das variáveis de entrada, buscando a melhoria da probabilidade da estrutura. Quando a adição de um nó pai ao nó não aumenta a probabilidade da estrutura, o algoritmo pára de adicionar nós pais a esse nó e escolhe a próxima variável para formar o próximo nó (busca subida de encosta). Deve ser usada uma função *Pred(x_i)* que retorna o conjunto de nós que precede x_i na ordenação dos nós. O pseudocódigo seguinte (adaptado de [15]), descrito na Figura 2.5, descreve o algoritmo K2.

```
1. Algoritmo K2:

2. {Entrada: Um conjunto de n nós, uma ordenação dos nós, limite superior u sobre o
3.            número de pais que um nó pode ter e uma base de dados D, contendo m casos.}
4. {Saída: Para cada nó, a identificação dos pais do nó.}
5.    for i := 1 to n do
6.        πi := {};
7.        Pold := g(i, πi); {Esta função é calculada usando a equação (4).}
8.        OKToProceed := true
9.        while OKToProceed and | πi | < u do
10           suponha z ser o nó em Pred(xi) - πi que maximiza g(i, πi, ∪ {z});
11.          Pnew := g(i, πi, ∪ {z});
12.          if Pnew > Pold then
13.              Pold := Pnew;
14.              πi := πi ∪ {z};
15.          else OKToProceed := false;
16.     end {while};
17.     write ('Nó:', xi, 'Pais deste nó:',  πi)
18. end {for};
19. end {K2};
```

Figura 2.5. Pseudocódigo do algoritmo K2, adaptado de [15].

A função (4) pode ser calculada no tempo *O(m u r)*, onde *m* é o número de casos da base de dados, *u* é o número máximo de pais que um nó pode ter, fornecido pelo usuário, e *r* é o número de valores possíveis da variável. Para melhorar a velocidade de execução do algoritmo, as funções *g(i, π_i)* e *g(i, $\pi_i \cup \{z\}$)* podem ser substituídas por *log(g(i, π_i))* e *log(g(i, $\pi_i \cup \{z\}$))*, respectivamente. Para melhor entendimento do algoritmo, é dado, na seqüência, um exemplo de aplicação do algoritmo K2 (Exemplo 1).

Exemplo 1 – Aprendizado da estrutura de uma rede *Bayesiana* utilizando K2

A Tabela 2.3 constitui uma base de dados, extraída do repositório da Universidade da Califórnia, em *Irvine* [46], e será aqui utilizada para o aprendizado da estrutura da rede *Bayesiana*, formada pelas variáveis: *age, spectacle-prescrip, astigmatism, tear-prod-rate, contact-lenses*. Cada linha da tabela, chamada de caso, representa uma instância de treinamento. Para realizar os cálculos, será usada a versão logarítmica da função dada pela equação (4).

Tabela 2.3. Base de dados para as variáveis age, spectacle-prescrip, astigmatism, tear-prod-rate, contact-lenses.

	Valores para cada variável				
Caso	***age***	***spectacle-pescrip***	***astigmatism***	***tear-prod-rate***	***contact-lenses***
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none
6	young	hypermetrope	no	normal	soft
7	young	hypermetrope	yes	reduced	none
8	young	hypermetrope	yes	normal	hard
9	pre-presbyopic	myope	no	reduced	none
10	pre-presbyopic	myope	no	normal	soft
11	pre-presbyopic	myope	yes	reduced	none
12	pre-presbyopic	myope	yes	normal	hard
13	pre-presbyopic	hypermetrope	no	reduced	none
14	pre-presbyopic	hypermetrope	no	normal	soft
15	pre-presbyopic	hypermetrope	yes	reduced	none
16	pre-presbyopic	hypermetrope	yes	normal	none
17	presbyopic	myope	no	reduced	none
18	presbyopic	myope	no	normal	none
19	presbyopic	myope	yes	reduced	none
20	presbyopic	myope	yes	normal	hard
21	presbyopic	hypermetrope	no	reduced	none
22	presbyopic	hypermetrope	no	normal	soft
23	presbyopic	hypermetrope	yes	reduced	none
24	presbyopic	hypermetrope	yes	normal	none

Considerando-se que as variáveis estão ordenadas de acordo com seu posicionamento na base de dados, tem-se a seguinte ordenação: *age, spectacle-prescrip, astigmatism, tear-prod-rate e contact-lenses,* que é passada como entrada para o K2. A primeira variável a ser analisada é *age*, tendo os seguintes valores: $r_1 = 3$ *(young, pre-presbyopic, presbyopic)*, $w_{11} =$

não tem pai (como é a primeira variável na ordem que foi passada, não há pais) e, portanto, $q_1 = 1$.

Calculando *Pold* de acordo com a função (4) na forma logarítmica:

$$Pold = log(g(i, \pi_i)) = ln((3-1)!) - ln((N_{ij} + 3 - 1)!) + \sum_{k=1}^{3} N_{ijk}!\,,$$

$$= ln(2!) - ln((24 + 3 - 1)!) + ln(8!) + ln(8!) + ln(8!)$$
$$= 0,69 - 61,26 + 10,60 + 10,60 + 10,60$$
$$= -28,75$$

Como a primeira variável não pode ter pais, o algoritmo encerra a primeira iteração do laço principal e segue para a segunda, onde analisará a próxima variável. A Figura 2.6 mostra a estrutura da rede *Bayesiana*, induzida até o momento.

Figura 2.6. Estrutura parcial da rede *Bayesiana,* após a primeira iteração do K2.

A segunda variável a ser analisada é *pectacle-pescrip.* Valores: $r_2 = 2$ (*myope, hypermetrope*). Ela pode ter a variável anterior, *age*, como pai e, portanto, terá $w_{22} = w_{21}$, w_{22} *e* w_{23}, pois a variável *age* tem três valores possíveis. w_{21} = *young,* w_{22} = *pre-presbyopic,* w_{23} = *presbyopic,* e $q_2 = 3$.

Calculando *Pold:* (inicialmente calcula-se o valor de *g* (função (4)) para a variável sem pai).

$$Pold = log(g(i, \pi_i)) = ln((2-1)!) - ln((N_{ij} + 2 - 1)!) + \sum_{k=1}^{2} N_{ijk}!,$$

$$= ln(1!) - ln((24 + 2 - 1)!) + ln(12!) + ln(12!)$$
$$= 0 - 58 + 19,98 + 19,98$$
$$= -18,02$$

O algoritmo K2 procura pelos possíveis pais da variável *spectacle-pescrip*. Na linha 10, mostrada na Figura 2.5, o K2 calcula o valor de *g* para a variável com os seus possíveis pais. Assim, *Pnew* será:

$$Pnew = log(g(i, \pi_i)) = \ln((2-1)!) - \ln((N_{ij} + 2 - 1)!) + \sum_{k=1} N_{ijk}!,$$

(N_{ijk} representará os valores dos atributos da variável *spectacle-pescrip* combinados com os atributos da variável *age* encontrados na base de dados. Por exemplo: v_{11} = *myope* e w_{11} = *young*, v_{12} = *hypermetrope*, w_{11} = *young*, e assim sucessivamente).

$$Pnew = log(g(i, \pi_i)) = [\ln(1!) - \ln((8 + 2 - 1)!) + \ln(4!) + \ln(4!)] + [\ln(1!) - \ln((8 + 2 - 1)!) + \ln(4!) + \ln(4!)] + [\ln(1!) - \ln((8 + 2 - 1)!) + \ln(4!) + \ln(4!)]$$
$$= [0 - \ln(9!) + 3{,}17 + 3{,}17] + [0 - \ln(9!) + 3{,}17 + 3{,}17] + [0 - \ln(9!) + 3{,}17 + 3{,}17]$$
$$= log(g(i, \pi_i)) = [-6{,}46] + [-6{,}46] + [-16{,}46]$$
$$= log(g(i, \pi_i)) = \mathbf{-\ 19{,}38}$$

O valor de *Pnew* para a variável *spectacle-pescrip,* tendo *age* como pai, é menor que o valor de *Pold*. Isto significa que a adição de um pai à variável não melhorou o *score* (pontuação) da rede. Como não há outras variáveis candidatas a ser pai de *spectacle-pescrip,* o algoritmo mantém a variável sem pais e encerra a iteração menor à procura de outro pai. Segue, então, para a terceira iteração do laço principal e analisará a terceira variável. A Figura 2.7 apresenta a nova estrutura, gerada após esta iteração.

Figura 2.7. Estrutura parcial da rede *Bayesiana,* após a segunda iteração do K2.

Para a variável *astigmatism* (terceira variável), existem duas candidatas a serem pais: *age* e *spectacle-prescrip*. O algoritmo calcula o valor de *g* para a variável sem pai: *-18,0292*. Na linha 10 da Figura 2.5, é calculado o valor de *g* para *astigmatism* tendo *age* como pai e depois tendo *spectacle-prescrip* como pai. Os valores obtidos são *-19,3372* e *-18,7873*, respectivamente. O melhor valor de *g* será o novo *Pnew*. Se *Pnew* for maior que *Pold*, significa que a adição de um pai à variável melhorou o *score*. Neste caso, o valor de *Pnew* foi

menor que *Pold*. A iteração em busca de novos pais será encerrada e o algoritmo irá procurar pelos pais da variável seguinte. Se *Pnew* fosse maior que *Pold*, o algoritmo iria calcular o valor de *g* para a variável *astigmatism* tendo dois pais.

O algoritmo analisa as variáveis restantes (*tear-prod-rate, contact-lenses*.), da mesma forma como analisou as anteriores, e no final gera a estrutura de rede *Bayesiana* mostrada na Figura 2.8, com *g = -100,671*:

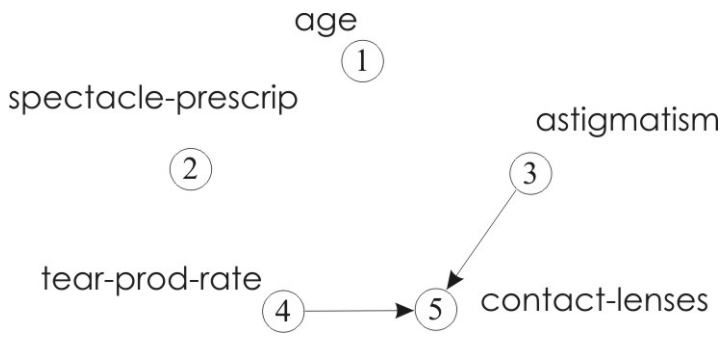


Figura 2.8. Rede encontrada pelo algoritmo K2 para as variáveis: *age, spectacle-prescrip, astigmatism, tear-prod-rate, contact-lenses*, nesta ordem de entrada.

A rede da Figura 2.8 é composta por 5 nós, correspondentes às variáveis de entrada, onde os nós *astigmatism* e *tear-prod-rate* estão conectados através de arcos direcionados ao nó *contact-lenses*. Os nós *astigmatism* e *tear-prod-rate* são considerados pais de *contact-lenses,* significando que possuem relação direta sobre este nó. A força dessas relações deve ser medida pelas probabilidades condicionais, que ainda serão definidas durante o aprendizado dos parâmetros numéricos (mais detalhes sobre o aprendizado dos parâmetros numéricos podem ser encontrados em [15]). Quanto aos outros nós sem conexão, pode-se dizer que não há relações diretas entre eles e os outros nós da rede. Vale lembrar que a estrutura de rede da Figura 2.8 foi gerada de acordo uma das possíveis ordenações das variáveis de entrada e que poderia ser diferente para uma outra ordenação. Por exemplo, se a ordenação das variáveis fosse dada por: *age, contact-lenses, astigmatism, spectacle-prescrip* e *tear-prod-rate,* o algoritmo K2 geraria a estrutura da Figura 2.9, com *g = -98,725*. Note que a Figura 2.9 apresenta os mesmos arcos que a Figura 2.8, porém invertidos, o que significa que,

dependendo da ordem de entrada das variáveis, pode-se ter uma rede melhor ou pior que a anterior.

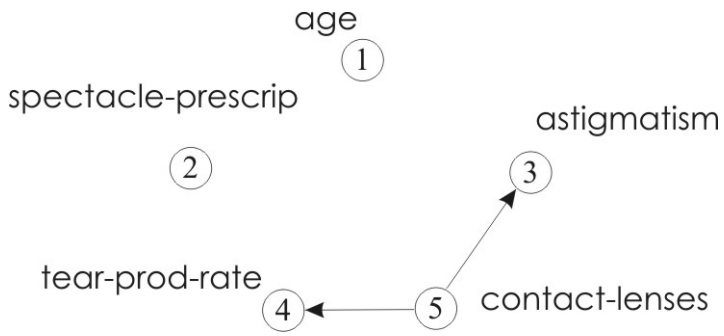


Figura 2.9. Rede encontrada pelo algoritmo K2 para as variáveis: *age, contact-lenses, astigmatism, spectacle-prescrip e tear-prod-rate*, nesta ordem.

2.2.2 Algoritmo PC

PC é um dos algoritmos que utiliza o conceito de independência condicional para a construção da estrutura de uma rede *Bayesiana*. Segundo [45], dado um conjunto de independências condicionais em uma distribuição de probabilidade, tenta-se encontrar um grafo direcionado acíclico (*gda*), onde a condição de *Markov* envolve todas e apenas estas independências condicionais. A condição de *Markov*, definida em [45], estabelece o seguinte:

Condição de Markov: *Seja G = (V,E) um gda, 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 ∈ 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 [45], 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 *gda* cujas d-separações sejam as mesmas que em IND_P.

A d-separação é uma propriedade do *gda*. Como definido em [45], seja $G = (V,E)$ um *gda,* $A \subseteq V$, 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 2.10, 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 2.11, 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.

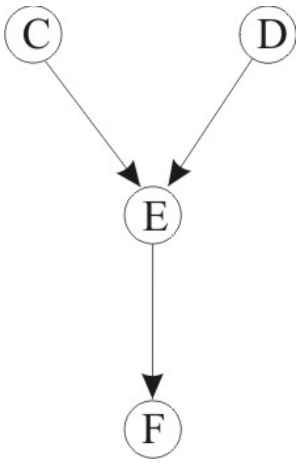


Figura 2.10. *gda* onde o conjunto de nós {C, D} é d-separado do nó F pelo nó E.

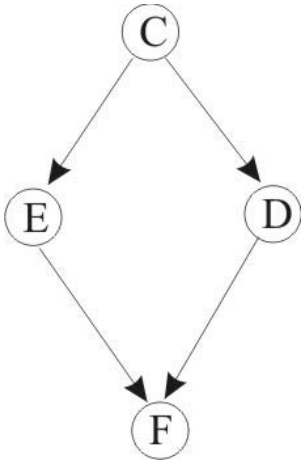


Figura 2. 11. ***gda*** **onde o nó** ***C*** **é d-separado do nó** ***F*** **pelos nós {*E, D*}.**

Um conjunto de d-separações *IND* no conjunto de nós *V* é um conjunto contendo declarações de d-separação para os membros de *V*. No pior caso, qualquer algoritmo teria que procurar 2^{n-2} subconjuntos de $V - \{X,Y\}$ para determinar se *X* e *Y* são d-separados. A razão é que, dado um subconjunto $U \subset V - \{X,Y\}$, *X* e *Y* não poderiam estar d-separados por *U,* mas d-separados por algum superconjunto ou subconjunto de *U*. Então, elimina-se um conjunto toda vez que este for testado para d-separação. Contudo, no caso de *gdas* esparsos, pode-se melhorar considerando pequenos subconjuntos primeiro. De acordo com [45], se *X* e *Y* são d-separados no *gda G*, então eles são d-separados pelos pais de *X* em *G* ou pelos pais de *Y* em *G*. Isto significa que é preciso considerar apenas subconjuntos de ADJ_X (adjacentes a *X*) em *gda* e subconjuntos de ADJ_Y (adjacentes a *Y*) em *gda* quando deseja-se determinar se *X* e *Y* são d-separados em *gda*. ADJ_X significa o subconjunto de *V* consistindo de todos os nós adjacentes a *X* (um nó é adjacente a outro se e somente se eles não são d-separados em *gda* e existe um caminho de *X* para *Y*).

O algoritmo, chamado de PC por seus desenvolvedores em [61], primeiro analisa os subconjuntos de ordem 0 (ou seja, os subconjuntos que possuem d-separacão 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 *gda* como a estrutura B_s de uma rede *Bayesiana*, este processo pode ser utilizado para induzir uma estrutura de rede *Bayesiana*.

A Figura 2.12, adaptada de [61], apresenta o algoritmo PC. O conjunto de variáveis condicionadas precisa 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 possui, como entradas, uma base de dados e um conjunto de d-separações *IND* de ordens 0, 1, 2, e subseqüentes.

A seguir, o exemplo 2 mostra como o algoritmo define B_s. A Figura 2.13 traça a operação das duas primeiras partes do algoritmo PC. O símbolo ├ é usado para representar d-separação. Por exemplo, para n=1, **A ├ C \B,** significando que o nó A é d-separado do nó C pelo nó B.

PC Algorithm:
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)\{Y}* tem cardinalidade maior ou igual a *n,* e um subconjunto *S* de *Adjacencies*(B_s, *X)\{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)\{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 -> Y <- Z,* se e somente se, *Y* não está em *Sepset(X, Z).*
D) **repete**
Se *A -> 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 ->C*.
Se há um caminho direcionado de A para B, e uma aresta entre A e B, então oriente A – B como A -> B.
enquanto não houver mais arestas a serem orientadas.

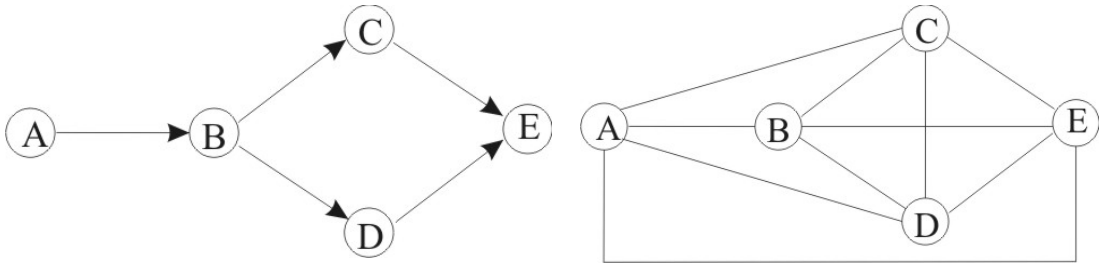
Figura 2.12. Pseudo-código do algoritmo PC [61].

Inicialmente, PC forma um grafo não direcionado completo B_s a partir do conjunto de nós *A, B, C, D* e *E*, conforme o estágio A) da Figura 2.12. De posse do conjunto de d-

separações *IND*, o algoritmo elimina as arestas de B_s, de acordo com o estágio B), iniciando pelos pares ordenados de variáveis adjacentes em B_s, de cardinalidade 0. Como não há d-separações de cardinalidade 0 em *IND*, não há nada a ser feito. Na próxima iteração, são analisados os pares de variáveis adjacentes em B_s, de cardinalidade 1.

Exemplo 2 – Aprendizado da estrutura de uma rede *Bayesiana* utilizando PC

Grafo Verdadeiro **Grafo Completo não direcionado**

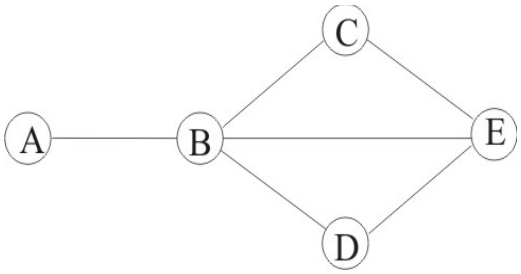


n = 0 Sem independências de ordem zero (sem d-separações).

n = 1 Independência de primeira ordem **Adjacências resultantes**

d-separações:

A ⊢ C \B
A ⊢ D \B
A ⊢ E \B
C ⊢ D \B



n = 2 Independência de segunda ordem **Adjacências resultantes**

d-separações:

B ⊢ E \ {C,D}

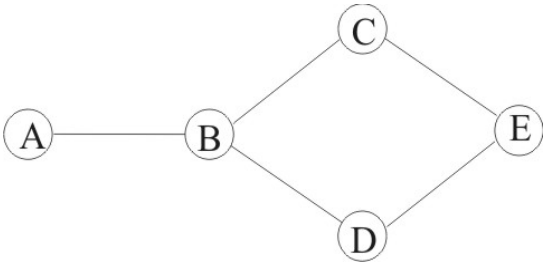


Figura 2.13. Exemplo de aplicação do algoritmo PC[61].

No conjunto de d-separações, tem-se que *A* é d-separado de *C*, dado *B*. Conforme o estágio B), a aresta de *A* a *C* é eliminada. Há ainda outras 3 d-separações que permitem eliminar as arestas de *A* a *D*, de *A* a *E* e de *C* a *D*. Na terceira iteração é eliminada a aresta de *B* a *E*, pois há uma d-separação de segunda ordem que diz que *B* é d-separado de *E*, dado as variáveis *C* e *D*. Visto que não há pares de variáveis adjacentes em B_s, de cardinalidade maior que 2, PC encerra o estágio B).

O grafo não direcionado, na Figura 2.13, estará parcialmente orientado no passo C). A tripla de variáveis, com apenas duas adjacências entre elas, são:

A – B – C; *A – B – D;*

C – B – D; *B – C – E;*

B – D – E; *C – E – D*

O nó *E* não está no *Sepset(C, D),* assim *C – E* e *E – D* colidem em *E*. Nenhuma das outras triplas formam colisões. O padrão final produzido pelo algoritmo é mostrado na Figura 2.14. Nota-se que alguns arcos não foram direcionados, pois o algoritmo nem sempre consegue ordenar todos os arcos. Neste caso, seria necessária outra maneira de direcioná-los, como, por exemplo, consultar um especialista, ou utilizar uma ordenação das variáveis.

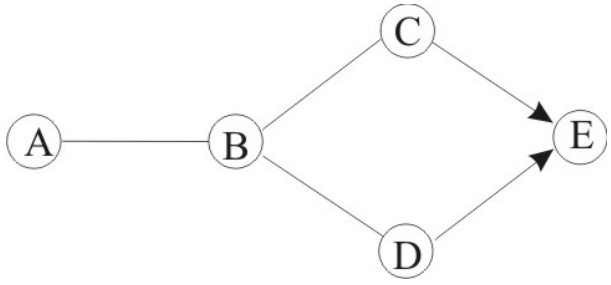


Figura 2.14. Padrão final produzido pelo PC para o exemplo anterior.

2.3 Aprendizado de Redes *Bayesianas* para Classificação (Classificadores *Bayesianos*)

A classificação é uma tarefa básica em análise de dados e reconhecimento de padrões e requer a construção de um classificador, ou seja, uma função que atribui uma variável classe a uma instância descrita por um conjunto de atributos. A indução de classificadores a partir de um conjunto de dados de instâncias pré-classificadas é um problema importante em aprendizado de máquina. Muitas abordagens para este problema são baseadas em várias representações, como árvores de decisão, listas de decisão, redes neurais, grafos de decisão e regras [20]. Redes *Bayesianas* também podem ser aplicadas na tarefa de classificação.

Redes *Bayesianas* oferecem uma solução para o problema de classificação discreta [9]. Um atributo discreto é um conjunto finito, por exemplo, o atributo *Pressão* pode ser definido como *Pressão = {baixa, media, alta}*. Um domínio discreto é um conjunto finito de atributos discretos. Suponha que $X = \{x_1, ..., x_m\}$ seja um domínio discreto, onde $x_1, ..., x_m$ são os atributos, como visto no início deste capítulo. Um domínio discreto classificado é um domínio discreto onde um dos atributos é distinguido como "classe" e pode ser representado como $X_c = \{a_1, ..., a_n, c\}$, onde $a_1, ..., a_n$, são considerados parte de um domínio discreto classificado e c é a classe.

Um classificador em um domínio discreto classificado X_c é um procedimento que, dado um conjunto de dados D em X_c e uma observação (instância) S em X_c, atribui uma classe a S.

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 (como visto na Seção anterior). 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.

Há, entretanto, alguns algoritmos específicos para construção de classificadores *Bayesianos* que distinguem a variável classe desde o início do processo de indução a partir dos dados.

Alguns destes classificadores *Bayesianos* têm mostrado bom desempenho, embora ainda tenham algumas desvantagens. Nas subseções seguintes, são apresentadas quatro abordagens de classificadores *Bayesianos,* conhecidas na literatura como: classificador *Bayesiano Naive*, TAN, KDB e Semi-*Naive*.

2.3.1 Classificador *Bayesiano Naive*

O classificador *Bayesiano Naive*, ou *Naive Bayes*, foi descrito pela primeira vez, em aprendizado de máquina, por *Cestnik et al* [10], embora já fosse conhecido na comunidade de reconhecimento de padrões [42].

A principal vantagem desta abordagem é o fato de que a estrutura (grafo) é sempre fixa: todos os atributos $A_i, ..., A_n$ são considerados independentes condicionalmente um do outro, dado o valor do atributo classe C. Desta forma, o processo de aprender o classificador é muito rápido, visto que a ordem de dependências a ser encontrada é fixa e reduzida a duas variáveis e, além disso, apenas os parâmetros numéricos precisam ser aprendidos.

Quando representado como uma rede *Bayesiana*, o *Naive Bayes* tem a estrutura descrita na Figura 2.15. Esta figura mostra que o atributo classe C é o único pai para cada atributo Ai, denotado como $\pi_{Ai} = \{C\}$, para todo $1 \leq i \leq n$. Por isso, a distribuição de probabilidade conjunta $P(A_1, ..., A_n \mid C)$ para esta rede é determinada por:

$$P(A_1, ..., A_n / C) = \prod_{i=1}^{n+1} P(U_i / \pi_{U_i} = P(C) \prod_{i=1}^{n} P(A_i / C) , \qquad (5)$$

e, da definição de probabilidade condicional, a probabilidade para as classes em C, dado os valores dos atributos, é:

$$P(C / A_1, ..., A_n) = \alpha P(C) \prod_{i=1}^{n} P(A_i / C), \qquad (6)$$

onde α é uma constante de normalização.

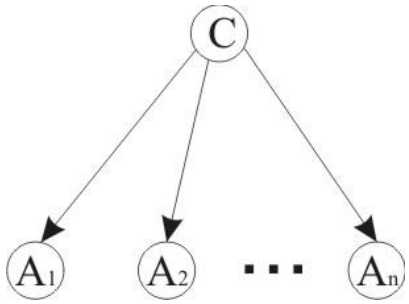


Figura 2.15. Estrutura de rede de um classificador *Bayesiano Naive.*

Como informado na literatura [54], o desempenho do *Naive Bayes* é surpreendentemente bom para muitos problemas de classificação, sendo capaz de obter resultados comparáveis a outros classificadores mais complexos. No entanto, faz fortes suposições de independência que, na prática, não acontecem, o que pode levar a uma estimativa de probabilidade pouco confiável.

2.3.2 Classificador TAN

Tree augmented naive Bayes (TAN) [20] é um classificador *Bayesiano,* onde o atributo classe não tem pais e, cada atributo, tem, como pai, o atributo classe e, no máximo, um outro atributo.

O classificador TAN considera as dependências entre os outros atributos, além do atributo classe *C.* Estes modelos representam as relações entre os atributos A_i, ..., A_n, condicionados ao atributo classe *C*, usando uma estrutura de árvore. A rede da Figura 2.16 representa um modelo TAN.

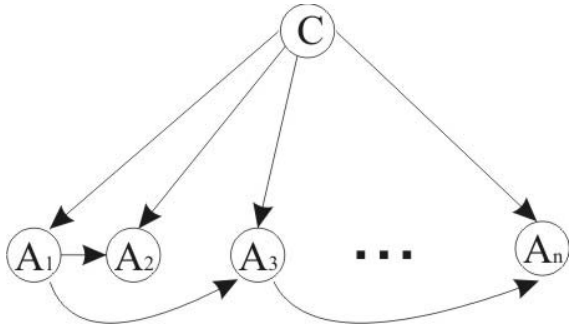


Figura 2.16. Estrutura de rede de um classificador TAN.

Uma rede TAN é inicializada como uma rede *Naive Bayes*. Os arcos adicionais entre os atributos são encontrados através de um algoritmo de busca.

O fato de haver, no máximo, mais um pai adicional para cada atributo significa que pode haver um arco no grafo da variável A_i para A_j. Isso implica que esses dois atributos, A_i e A_j, não são independentes, dado a classe. A influência de A_j nas probabilidades da classe depende também do valor de A_i.

2.3.3 Algoritmo KDB

Sahami [54] apresentou um algoritmo, chamado KDB, que tem como entrada um conjunto de dados de instâncias pré-classificadas, DB, e um valor k para o grau máximo permitido de dependência da variável. A saída de KDB é um classificador *k-dependence Bayesian*, com tabelas de probabilidade condicional, determinadas a partir dos dados de entrada.

Um classificador *k-dependence Bayesian*, de acordo com [54], é uma rede *Bayesiana* que contém a estrutura do classificador *Bayesiano Naive* e permite que cada variável X_i tenha no máximo *k* nós como pais. Pela proposição 1 de [54], o classificador *Bayesiano Naive* é um classificador *0-dependence Bayesian*. E pela proposição 2 de [54], um classificador *Bayesiano*

completo (ou seja, sem independências) é um classificador *(N-1)-dependence Bayesian*, onde *N* é o número de variáveis no domínio.

KDB foi definido pelo autor como um espectro de dependências permitidas em um modelo probabilístico; com o algoritmo *Naive Bayes*, mais restrito de um lado, e o aprendizado de redes *Bayesianas* do outro. Por variar o valor de *k*, KDB pode definir modelos que suavemente move-se ao longo do espectro de dependências da variável.

Uma característica importante de KDB, que o torna adequado para o domínio de mineração de dados, é sua pequena complexidade computacional. Assim, torna-se uma alternativa aos algoritmos de aprendizado de redes *Bayesianas*, que fazem uma busca custosa através do espaço de estruturas de redes ou de dependências de variáveis. Além disso, permite capturar muito da eficiência do modelo *Bayesiano Naive*.

2.3.4 Classificador Semi-*Naive*

A abordagem semi-*Naive* foi proposta por *Kononenko* [32] e pode ser considerada como um tipo mais sofisticado de classificador *Bayesiano*, com relação ao tipo de dependências que é levado em consideração e a maneira como permite que grupos de atributos sejam considerados como um único nó na rede *Bayesiana*.

O objetivo do classificador semi-*Naive* é evitar as premissas restritas do *Naive Bayes*, permitindo agrupar alguns atributos em um único nó da estrutura. A Figura 2.17 ilustra um exemplo de classificador semi-*Naive* em um problema com quatro atributos, mostrando que a estrutura de rede *Bayesiana* trata alguns atributos agrupados como um único atributo, de acordo com a fatorização da distribuição de probabilidade.

O classificador semi-*Naive* é um modelo da forma:

$$P(C / A_1, ..., A_n) = P(C) \, . \, P(A_1 / C) \, ... \, P(A_k / C), \qquad (7)$$

onde $A_1, ..., A_n$ são grupos disjuntos de atributos. Tal modelo assume que A_i é

condicionalmente independente de A_j se, e somente se, eles estão em grupos diferentes. Assim, nenhuma suposição de independência é feita sobre atributos que estão no mesmo grupo.

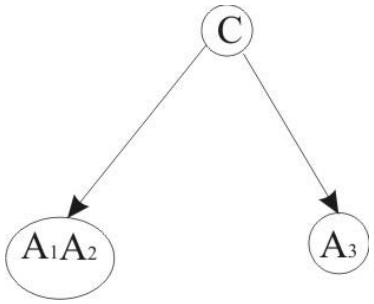


Figura 2.17. Estrutura de rede de um classificador semi-*Naive.*

O método de *Kononenko* usa testes estatísticos de independência para particionar os atributos em grupos. Esse procedimento, no entanto, tende a selecionar grupos grandes, o que pode levar a problemas de *overfitting*.

2.4 Ordenação de Variáveis

Muitos algoritmos de aprendizado baseados na busca heurística têm como importante característica a dependência do processo de aprendizagem em relação à ordenação das variáveis [61]. Esta dependência ocorre da seguinte forma. Considere uma base de dados com três atributos A1, A2 e A3. Se o processo de aprendizado for realizado com os atributos ordenados desta maneira, isto implica que A1 não terá pais na rede, pois é o primeiro atributo, mas poderá ser pai de A2 e A3; já A2 poderá ter como pai A1 e poderá ser pai de A3; e finalmente A3 não terá filhos, mas poderá ter como pais A1 e A2. 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, como no exemplo 2 (apresentado no final da Seção 2.2.2, exemplificando o algoritmo PC), 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. Na rede descrita na Figura 2.14, poderia ter sido usada, por exemplo, a ordem alfabética das variáveis como ordenação prévia, assim todos os seus arcos seriam direcionados, como mostra a Figura 2.18. Embora não exijam uma ordenação prévia, estes algoritmos são mais rápidos quando podem contar com tal ordenação, e, como foi mostrado em [30], possuem uma tendência de melhores resultados quando aplicados com uma boa ordenação prévia das variáveis.

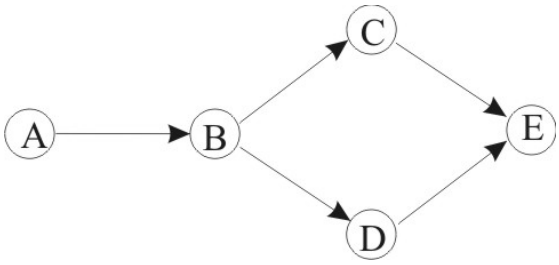


Figura 2.18. Padrão final produzido pelo PC para o exemplo anterior, utilizando a ordem alfabética das variáveis como ordenação prévia (A, B, C, D, E).

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 da geração de uma rede bayesiana, uma ordem que não represente de forma correta

o relacionamento das variáveis pode inserir erros no processo de aprendizagem. Como exemplo, observe a rede da Figura 2.19 (adaptada de [49]).

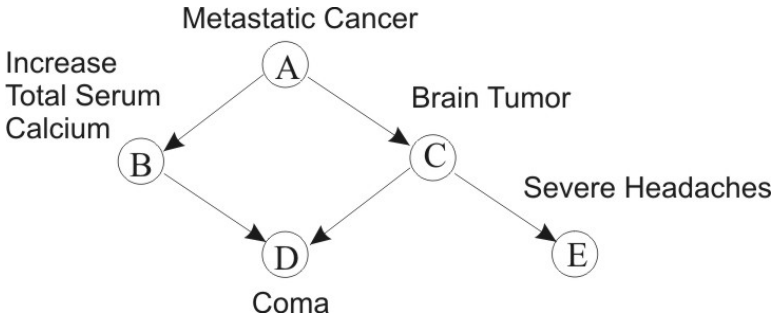


Figura 2.19. Rede *Bayesiana* representando o problema do câncer metastático (adaptada de [49]).

Supondo que será utilizada uma ordenação das variáveis para a construção da rede, então a variável A deve anteceder as variáveis B e C; a variável B deve anteceder a variável D; a variável C deve anteceder as variáveis D e E. Assim, uma possível ordenação, que permite a identificação da estrutura apresentada na Figura 2.19, seria A, B, C, D, E. Por outro lado, se a ordenação dada fosse E, D, C, B, A, então o direcionamento dos arcos seria como mostrado na Figura 2.20. Portanto, nota-se que a ordenação incorreta pode impedir a definição da rede correta.

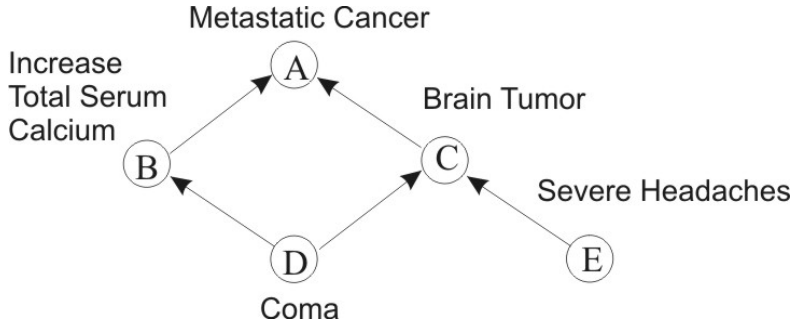


Figura 2.20. Possível rede *Bayesiana* representando o problema do câncer metastático com ordenação E, D, C, B, A.

Este trabalho enfatiza a busca de uma ordenação adequada para o aprendizado supervisionado de redes *Bayesianas*, usadas para classificação, visto que não existem métodos automáticos computacionalmente eficientes na resolução deste problema. Uma alternativa adequada em problemas de busca é o uso de técnicas de computação evolutiva, que será abordada no capítulo seguinte.

3 ALGORITMOS GENÉTICOS

No início da década de 50, alguns pesquisadores buscaram inspiração na Teoria da Evolução Natural e na Genética para criar modelos computacionais que dariam origem à área de pesquisa nomeada de Computação Evolutiva ou Evolucionária.

A computação 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. 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 algoritmos evolucionários variam conforme os vários modelos de computação evolutiva existentes, que pode ser dividida em 3 principais categorias, de acordo com [2]: Algoritmos Genéticos, Estratégias de Evolução e Programação Genética. As principais diferenças estão na representação dos indivíduos e, consequentemente, 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.

A Figura 3.1 mostra um algoritmo típico [23] que descreve os principais passos de qualquer algoritmo evolucionário. Uma introdução mais detalhada sobre algoritmos evolucionários, pode ser encontrada em [2], [59], [41], [42] e [58].

Neste capítulo, serão descritas as principais características dos algoritmos genéticos, destacando-os como boas ferramentas de busca, os quais serão aplicados na identificação de uma ordenação de variáveis adequada ao processo de aprendizado de redes *Bayesianas*.

```
Algoritmo 1: Algoritmo Evolucionário
  t = 0;
  Gerar População Inicial P(t);
  avaliar (P(t));
  enquanto Critério de parada não for satisfeito faça
      P'(t) = selecionar (P(t));
      P''(t) = aplicar_operadores_genéticos (P'(t));
      P(t+1) = criar_população_seguinte (P(t), P''(t));
      avaliar (P(t + 1));
     t = t + 1;
  fim enquanto
Devolver melhor indivíduo;
```

Figura 3.1. Pseudo-código [23] do algoritmo evolucionário típico.

3.1 Introdução aos Algoritmos Genéticos

Os algoritmos genéticos foram introduzidos por *John Holland* [27] e popularizados por um dos seus alunos, *David Goldberg* [21]. Estes algoritmos constituem uma das mais conhecidas classes de algoritmos evolucionários e partem do pressuposto que indivíduos com boas características genéticas têm maiores chances de sobreviver e produzir indivíduos mais aptos em uma dada população.

Os algoritmos genéticos 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 ferramenta de busca e otimização para a solução dos mais diferentes tipos de problemas [21].

Toda tarefa de busca ou otimização possui vários componentes, como o espaço de busca, onde são consideradas todas as possibilidades de solução de um determinado problema, e a função de aptidão (também conhecida como função de *fitness* ou função de avaliação), que é uma maneira de avaliar os elementos do espaço de busca. Existem diversos métodos de busca e funções de aptidão.

Os algoritmos genéticos diferem dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos [21]:

1. trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros;
2. trabalham com uma população e não com um único ponto;
3. utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;
4. utilizam regras de transição probabilísticas e não determinísticas.

Os algoritmos genéticos empregam uma estratégia de busca paralela e estruturada, embora aleatória, direcionada à busca de pontos de "alta aptidão", ou seja, pontos nos quais a função a ser minimizada ou maximizada tem valores relativamente baixos ou altos (respectivamente). Apesar de aleatórios, algoritmos genéticos não são buscas aleatórias não direcionadas, pois exploram informações históricas para encontrar novos pontos de busca onde são esperados melhores desempenhos. Isso é feito por meio de processos iterativos, nos quais cada iteração é chamada de geração.

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 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 de mutações e cruzamento (ou *crossover*), ou recombinação genética, 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.

Durante cada geração, os princípios de seleção e reprodução são aplicados a uma porcentagem de candidatos que pode variar, dependendo da complexidade do problema e dos recursos computacionais disponíveis. As iterações são repetidas até que seja encontrada uma solução satisfatória, de acordo com o critério de parada. Embora possam parecer simplistas do ponto de vista biológico, esses algoritmos são suficientemente complexos para fornecer mecanismos poderosos e robustos de busca adaptativa [51].

A Figura 3.2, adaptada de [51], apresenta o diagrama geral de um algoritmo genético, resumindo o processo descrito. Mais específico que o algoritmo da Figura 3.1, o algoritmo da Figura 3.2 aplica os operadores genéticos de cruzamento e mutação. Nas seções seguintes, serão apresentados, com mais detalhes, a representação do problema, os métodos de seleção e os operadores de cruzamento e mutação.

```
Algoritmo 2: Algoritmo Genético
1:  t = 0;
2:  Gerar População Inicial P(t);
3:  para cada indivíduo i da população atual P(t) faça
4:      Avaliar aptidão do indivíduo i;
5:  fim para
6:  enquanto Critério de parada não for satisfeito faça
7:      t = t + 1;
8:      Selecionar população P(t) a partir de P(t-1);
9:      Aplicar operadores de cruzamento sobre P(t);
10:     Aplicar operadores de mutação sobre P(t);
11:     Avaliar P(t);
12: fim enquanto
```

Figura 3.2. Pseudo-código do algoritmo genético (adaptada de [51]).

3.2 Representação

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 algoritmos genéticos 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 são genotípicas, 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 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 fenotípicas, facilitariam seu uso em determinados ambientes. Neste caso, precisam ser criados os operadores específicos para utilizar essas representações [51].

Neste trabalho, os algoritmos evolutivos propostos para buscar uma ordenação de variáveis adequada ao processo de aprendizado de redes *Bayesianas*, utilizam números inteiros positivos para representar as variáveis da base de dados. Por exemplo, a rede *Bayesiana* da Figura 2.1, possui as seguintes variáveis: x_1, x_2, ..., x_6; estas variáveis podem ser representadas pelos números inteiros positivos 1, 2, ..., 6, respectivamente. Um possível cromossomo para este domínio seria representado por [1, 3, 2, 5, 4, 6]. Com este tipo de representação, os operadores de cruzamento (Seção 3.3) não criam informações novas (ou seja, novos números inteiros) e, portanto, não modificam o número de variáveis da rede.

3.3 Seleção

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

O algoritmo genético começa com uma população inicial de *N* cromossomos. À cada cromossomo da população é atribuído um valor dado por uma função f_{apt} denominada função de aptidão. Esta função recebe, como entrada, os valores do gene do cromossomo e fornece, como resultado, a sua aptidão.

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. Na seqüência, são apresentados três métodos de seleção.

3.3.1 Método da Roleta

O método da roleta é o método de seleção mais simples e também o mais utilizado [51]. Cada indivíduo da população é representado na roleta por uma fatia proporcional ao seu índice de aptidão. Assim, indivíduos com maiores aptidões ocupam fatias maiores da roleta. A Figura 3.3 ilustra a criação de uma roleta a partir dos valores de aptidão dos cromossomos S_1, S_2, S_3, S_4 e S_5 de uma população, supondo que estes cromossomos representem possíveis soluções de um problema.

Para a seleção dos indivíduos, a roleta é girada um determinado número de vezes,

definido pelo tamanho da população. A cada giro, o indivíduo apontado pela seta ou agulha ilustrada na figura é selecionado. Os indivíduos selecionados são inseridos na população intermediária.

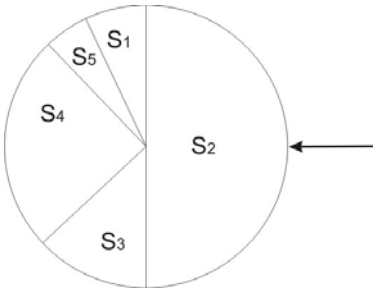


Figura 3.3. Método de seleção por roleta.

3.3.2 Método do Torneio

Neste caso, são escolhidos aleatoriamente (com probabilidades iguais) *n* cromossomos da população e o cromossomo com maior aptidão, entre esses, é selecionado para a população intermediária. O processo se repete até que a população intermediária esteja preenchida. Utiliza-se geralmente o valor *n=3* [51], mas este valor pode ser alterado, dependendo do problema de aplicação. A Figura 3.4 ilustra o método de seleção por torneio para os cromossomos S_1, S_2, S_3, S_4 e S_5.

Cromossomos Candidatos	⇨	Cromossomo Selecionado
S_1, S_2, S_5	⇨	**S_2**
S_2, S_4, S_5	⇨	**S_2**
S_5, S_1, S_3	⇨	**S_1**
S_4, S_5, S_3	⇨	**S_4**
S_3, S_1, S_5	⇨	**S_1**

Figura 3.4. Método de seleção por torneio.

3.3.3 Método da Amostragem Universal Estocástica

É uma variação do método da roleta em que, em vez de uma única agulha, *n* agulhas igualmente espaçadas, são utilizadas, onde *n* é o número de indivíduos a serem selecionados para a próxima geração. Assim, em vez de *n* vezes, a roleta é girada uma única vez, exibindo menos variância que as repetidas chamadas do método da roleta. Este método é ilustrado na Figura 3.5 para os cromossomos S_1, S_2, S_3, S_4 e S_5.

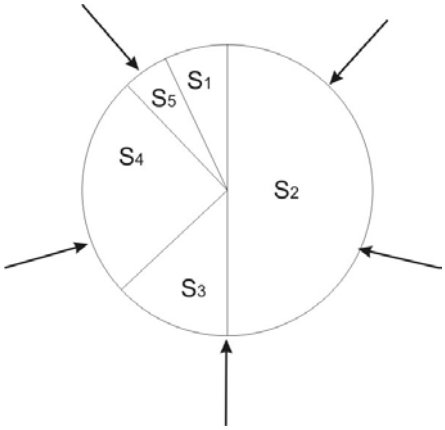


Figura 3.5. Método de seleção pela amostragem universal estocástica.

3.4 Operadores Genéticos

Os algoritmos genéticos utilizam um conjunto de operadores para gerar sucessivas populações, com melhores aptidões, 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 (*Pm*), que geralmente é uma taxa pequena, por exemplo $(0{,}01 \leq Pm \leq 0{,}1)$ [51], pois é um operador

secundário. A Figura 3.6 apresenta um exemplo de aplicação do operador de mutação em um cromossomo com codificação binária.

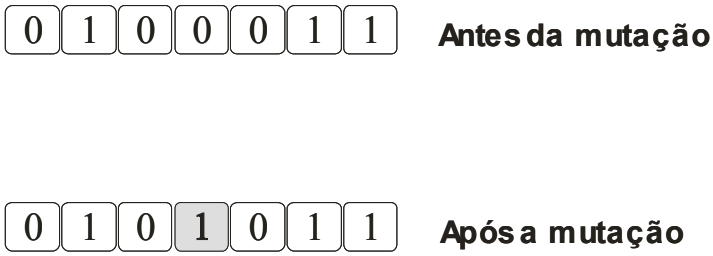


Figura 3.6. Exemplo de mutação.

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 ≤ Pc ≤ 0,99)* [51], que deve ser maior que a taxa de mutação. Este operador pode ser utilizado de várias maneiras:

a) **Cruzamento de um ponto:** Um ponto de cruzamento é escolhido de maneira aleatória e, a partir dele, as informações genéticas dos pais serão trocadas. Veja o exemplo da Figura 3.7 onde o cruzamento de um ponto é aplicado aos cromossomos pais A e B, produzindo os cromossomos filhos C e D.

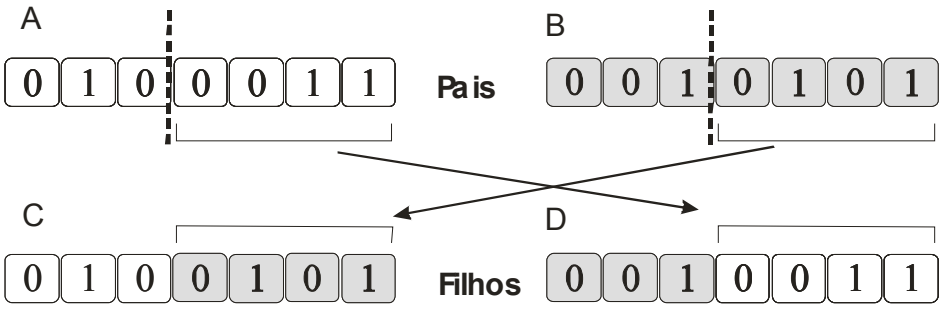


Figura 3.7. Exemplo de cruzamento de um ponto.

b) Cruzamento multipontos: É uma generalização da idéia de troca de material genético através de pontos, em que vários pontos de cruzamento podem ser utilizados. O funcionamento deste operador pode ser visto na Figura 3.8, para o caso de 2 pontos, nos cromossomos pais A e B, produzindo os cromossomos filhos C e D.

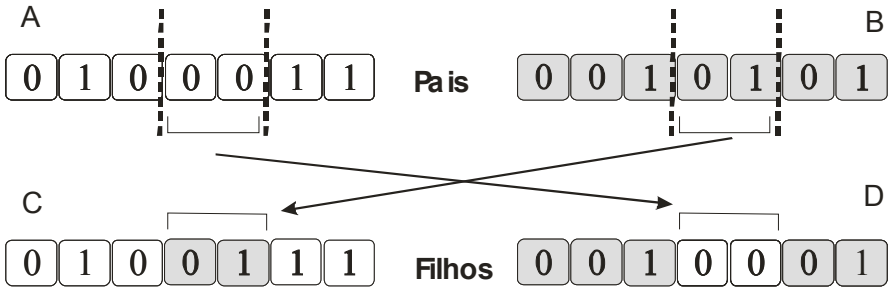


Figura 3.8. Exemplo de cruzamento de dois pontos.

c) Cruzamento uniforme: Não utiliza pontos de cruzamento, mas determina, por meio de uma máscara, os genes de cada cromossomo pai que cada filho herdará. Um exemplo da troca de informações provocada por este operador pode ser visto na Figura 3.9, sendo aplicado aos cromossomos pais A e B e gerando os cromossomos filhos C e D, codificados de forma binária. Um valor 1 na máscara indica que o gene correspondente do pai A será herdado pelo filho C, e o gene do pai B será herdado pelo filho D. Para um valor igual a 0 na máscara, ocorre o inverso.

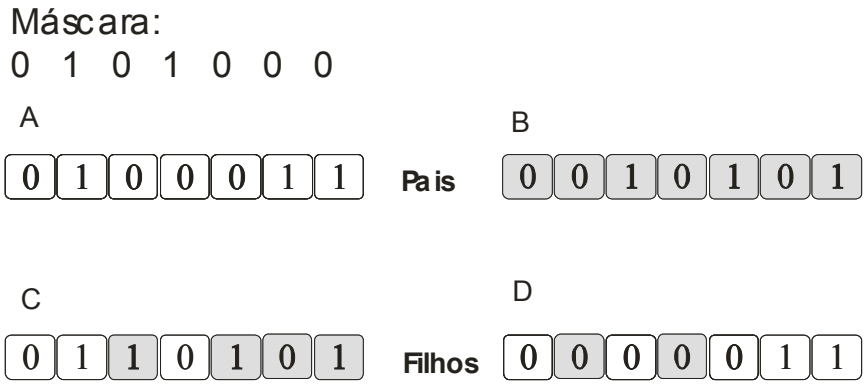


Figura 3.9. Exemplo de cruzamento uniforme.

3.5 Operadores Genéticos para Permutações

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 [33].

Uma permutação de *m* elementos é uma seqüência de *m* elementos em que nenhum elemento é repetido. Por exemplo, (*A*, *B*, *C*) e (*C, A, B*) são exemplos de duas permutações dos elementos *A, B e C*. Um grande número de problemas de otimização combinatória pode ter suas soluções representadas por permutações [33].

A ordenação das variáveis é um bom exemplo de permutação. Como descrito na Seção 2.3, a ordenação das variáveis pode influenciar o aprendizado de redes *Bayesianas*. A alternativa mais simples para se encontrar uma ordenação adequada seria a realização de uma busca exaustiva, verificando todas as ordenações possíveis. No entanto, isso não seria eficiente para problemas com um grande número de variáveis. Uma maneira de resolver a busca de uma ordenação adequada para o aprendizado de redes *Bayesianas*, usando algoritmos genéticos, é representar cada possível ordenação como um cromossomo.

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 na ordem, dois elementos do cromossomo são escolhidos aleatoriamente e suas posições são trocadas. A mutação por baralhamento 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 [21] e [63], são: *OBX (Order-Based Crossover), PBX (Position-Based Crossover), PMX (Partially Matched Crossover), CX (Cycle Crossover)* e *OX (Order Crossover).* Neste

trabalho, os métodos desenvolvidos utilizaram o operador *OX* (escolhido aleatoriamente) para realizar o cruzamento dos cromossomos; a seguir, é explicado o seu funcionamento, de acordo com [33].

OX (Order Crossover): O *crossover* OX inicia com dois cortes escolhidos aleatoriamente.

Pai_1:	A	B	C	D	F	E	G
Pai_2:	C	E	G	B	D	F	A

Os elementos de Pai_1 que correspondem aos elementos da sub-lista entre os dois cortes de Pai_2 são deletados.

Pai_1: A - | C - F | E -

Em seguida, os elementos situados na sub-lista de Pai_1 que não foram deletados são movidos de modo que todos os elementos fiquem juntos a partir do segundo corte, ou seja:

Pai_1: C F | - - - | E A

Finalmente, os espaços vazios são substituídos pela sub-lista entre os dois cortes de Pai_2, resultando no $Filho_1$:

$Filho_1$: C F | G B D | E A

Por processo similar é obtido $Filho_2$:

$Filho_2$: G B | C D F | A E

Vale notar que, no *crossover* OX, o que conta é a ordem dos elementos e não sua posição.

O algoritmo genético, proposto neste trabalho, utiliza o operador de mutação baseado na posição e o operador de cruzamento *OX*. Estes operadores foram escolhidos por serem

operadores definidos para permutação.

3.6 Parâmetros Genéticos

O desempenho de um algoritmo genético 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 [51]. 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 AGs. Em uma população pequena, o AG 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 AG pode se tornar 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 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 AG, por exemplo, parar após um certo número de gerações consecutivas em que não se obtém aperfeiçoamento da solução.

Para efeito de maior compreensão da utilização de algoritmos genéticos, o Apêndice A apresenta um exemplo didático de aplicação de algoritmos genéticos, tendo como objetivo a maximização da função $f(x) = x^3$.

No próximo capítulo, são apresentadas algumas abordagens, encontradas na literatura, que aplicam algoritmos genéticos na otimização do processo de aprendizado de redes *Bayesianas.*

4 MÉTODOS HÍBRIDOS PARA OTIMIZAÇÃO DO APRENDIZADO DE ESTRUTURA

Como mencionado anteriormente, o aprendizado da estrutura, *Bs*, de uma rede *Bayesiana* não é uma tarefa trivial, pois o espaço de busca para uma rede com *n* variáveis tem dimensão exponencial. Assim, encontrar a melhor estrutura, que representa as dependências entre as variáveis, é caracterizado como um problema NP [12][13], sendo difícil identificar a melhor solução para todos os domínios de aplicação.

Nos últimos anos, bons resultados alcançados têm motivado o desenvolvimento de muitos algoritmos de aprendizado [28]. Vários trabalhos na literatura propõem métodos híbridos, usando algoritmos evolucionários – apresentados no Capítulo 3 – para induzir redes *Bayesianas* a partir de dados. Nas seções seguintes, são apresentadas algumas destas abordagens. Há autores que se dedicam a domínios de aplicação específicos; outros focam no aprendizado de redes *Bayesianas* irrestritas; já alguns concentram-se no problema da ordenação de variáveis, como restrição imposta em busca de um menor esforço computacional.

4.1 Métodos Híbridos para Otimização do Aprendizado de Estrutura em Domínios de Aplicação Específicos

Uma aplicação no domínio de Biologia Molecular, usando algoritmos evolucionários e redes *Bayesianas*, é proposta em [40], na qual é apresentado um método de aprendizado de estrutura que simplifica uma rede *Bayesiana* induzida para detectar lugares de junção em seqüência de genes. Os autores aplicam algoritmos de seleção de atributos para reduzir a complexidade da rede *Bayesiana*. O método proposto utiliza um algoritmo genético para seleção de atributos, buscando um vetor binário, no qual todo *bit* é associado a um atributo no

espaço de atributos. Se um *bit* é 0, o atributo correspondente não será selecionado; caso contrário, o atributo é selecionado. Os resultados do trabalho revelam que a arquitetura da rede *Bayesiana* otimizada pelo método sugerido é mais adequada para predição de lugares de *Donor* (nome dado a um dos lugares de junção).

Outra aplicação, no domínio de Biologia Molecular, é apresentada em [66]. Neste trabalho, um algoritmo genético é aplicado no aprendizado de estrutura de redes *Bayesianas*, projetadas para representar uma rede de genes. O trabalho é voltado especialmente para reconstrução de redes de genes com *Yeast cell-cycle.*

Em [64], os autores focam no uso de redes *Bayesianas* e algoritmos evolucionários para conseguir uma explicação automática de séries temporais multivariadas (MTS – *multivariate time series*), através do aprendizado de redes *Bayesianas* dinâmicas (DBNs – *dynamic Bayesian networks*). Neste sentido, o desenvolvimento do algoritmo evolucionário leva em consideração certas características de MTS, em geral, para criar boas redes, tão rápido quanto possível. O método proposto é adequado para aprender DBNs a partir de MTS, com grandes tempos de atrazo, especialmente em situações de demanda de tempo.

Na área de mineração de dados, tem sido crescente o interesse em redes *Bayesianas* para representar conhecimento [69], porque podem manipular conjunto de dados incompletos e facilitar a combinação do domínio e dos dados. Diante dos problemas de aprendizado de redes *Bayesianas*, *Wong et. al.* [69] propuseram um novo algoritmo de mineração de dados que aplica coevolução cooperativa e uma abordagem híbrida para aprender redes *Bayesianas* irrestritas a partir de dados. A Seção 4.2, que trata de métodos híbridos para redes *Bayesianas* irrestritas, apresenta este algoritmo.

4.2 Métodos Híbridos para Otimização do Aprendizado de Estrutura em Redes *Bayesianas* Irrestritas

Alguns trabalhos concentram-se no aprendizado de redes *Bayesianas* irrestritas (isto é, aprendizado de redes *Bayesianas* que não são necessariamente criadas para problemas de classificação). Um algoritmo, proposto por *Wong et al* [69], foi chamado de CCGA (*Cooperative Coevolution Genetic Algorithm*). A idéia principal neste método é combinar as características das abordagens de análise de dependências e de busca e pontuação. A análise de dependências é conduzida na primeira fase para reduzir o tamanho do espaço de busca, enquanto um AG de coevolução cooperativa é usado na segunda fase para gerar as redes *Bayesianas*.

O algoritmo foi aplicado a um conjunto de dados do mundo real, usado para tomadas de decisão, e comparado com redes neurais *backpropagation* e modelos de regressão. Para este conjunto de dados, as redes *Bayesianas* tiveram melhores resultados que os outros modelos. O estudo mostrou que CCGA pode se tornar uma ferramenta poderosa e eficiente em mineração de dados. O trabalho descrito em [70] estende a idéia proposta em [69] e introduz um novo operador, buscando melhorias na eficiência e eficácia do algoritmo.

Campos, L. M. e *Huete, J. F* [6] consideram um subgrupo do conjunto de relações de dependências/independências para obter a ordenação de variáveis. Este processo é guiado por algoritmo genético e *simulated anneling*. A idéia básica é usar apenas um subconjunto de relações de dependências/independências do modelo para aprender uma parte da rede e, depois, aplicar uma ferramenta de otimização combinatorial para a busca de ordenação, preservando o máximo possível dessas dependências/independências. Os resultados obtidos em [6] mostram a validade da abordagem.

Em [57], um algoritmo genético semântico (SGA – *semantic genetic algorithm*) é

proposto para aprender a estrutura de redes *Bayesianas* a partir de dados. A geração da população inicial, os operadores de cruzamento e de mutação são os focos principais deste trabalho. Os autores propõem dois operadores semânticos de cruzamento e mutação, projetados para produzir uma convergência mais rápida. As estruturas semânticas das redes *Bayesianas* são incorporadas aos operadores de cruzamento e mutação e podem melhorar o desempenho do SGA, quando comparado a algoritmos genéticos clássicos para aprendizado de redes *Bayesianas*, em domínios específicos.

O trabalho apresentado em [38] propõe a união de uma abordagem híbrida usada para o aprendizado de estruturas de redes *Bayesianas*, através de programação evolucionária (HEP) [68], com o conceito de tamanho de população dinâmica de um algoritmo genético, baseado em população elitista adaptativa (AEGA – *Adaptive Elitist-population Based Genetic Algorithm*) [39]. HEP busca a estrutura de rede com a ajuda de informação de dependência estatística e usa o *score* MDL [34] como função de aptidão. Sua união com AEGA resultou em uma versão estendida do HEP, chamada A-HEP, a qual também aplica o *score* MDL como função de aptidão e adota o conceito de população dinâmica de AEGA.

Larrañaga P. et al [36] propuseram um algoritmo genético para encontrar a melhor estrutura de rede *Bayesiana*, a partir de uma base de dados. No trabalho, a estrutura da rede (composta de *n* nós) é representada por uma matriz de adjacências, *n x n*, *C*. Cada elemento na matriz é definido como:

$$C_{ij} = \begin{cases} 1, \text{ se o nó } j \text{ é um pai do nó } i \\ 0, \text{ caso contrário.} \end{cases}$$

Com esta representação, a *i-ésima* linha na matriz codifica o conjunto de pais do nó N_i (isto é, define o conjunto π_{N_i}. Um algoritmo genético simples (com um cruzamento de um ponto e mutação) é aplicado na busca pela melhor solução descrita na representação *bit-string*. Como

função de aptidão, foi adotada o *score Bayesiano* do algoritmo K2. Note que os operadores genéticos poderiam gerar estruturas ilegais (ou seja, redes que não são grafos direcionados acíclicos). Por isso, era necessário reparar ciclos, após a produção de uma descendência. O número de pais possíveis para um nó foi limitado na sua implementação. Apesar destas restrições reduzirem o espaço de busca, o problema ainda é NP.

4.3 Métodos Híbridos que buscam uma Ordenação de Variáveis Adequada para Otimização do Aprendizado de Estrutura

Encontrar uma ordenação adequada para as variáveis em uma rede *Bayesiana* é um problema complexo que, para ser resolvido, necessita de muita informação sobre o domínio do problema. Em geral, um especialista poderia disponibilizar este tipo de informação, mas, nem sempre, esse especialista está disponível. Portanto, tornou-se interessante desenvolver ferramentas capazes de obter informações que auxiliam no processo de aprendizado.

Larrañaga et al. [35] consideraram o problema de encontrar as ordenações das variáveis para o aprendizado. Os autores buscam uma boa ordenação entre as variáveis com um algoritmo genético, que cria novas ordenações por meio da aplicação de operadores de cruzamento e mutação. Em cada geração, duas ordenações são selecionadas para cruzamento, sendo que a probabilidade de uma ordenação ser selecionada depende de sua aptidão. A aptidão de cada ordenação é dada pela avaliação da estrutura da rede *Bayesiana* criada pelo algoritmo K2. Para isso, é usada a fórmula de *Cooper and Herskovitz* [15] (função *g*, apresentada na Seção 2.2.1).

Os autores estudaram o comportamento do algoritmo descrito com diferentes combinações de operadores de cruzamento e mutação. No entanto, não restringiram as redes *Bayesianas* criadas para classificação, como é proposto neste trabalho.

Hsu, W. H. [31] implementou um algoritmo genético para o problema de ordenação de variáveis no aprendizado de redes *Bayesianas* e para o problema de inferência. O autor destaca o uso do algoritmo K2 e o problema da ordenação das variáveis que serão passadas como entrada. Ele ainda ressalta que, para determinar a otimização da ordenação de variáveis como um problema de busca, devem ser definidos os estados (ordenações), operadores (re-ordenação), candidatos iniciais e critério de avaliação.

O algoritmo genético criado em [31] aplica o operador de cruzamento *OX* (Seção 3.5) e, como função de aptidão, calcula a perda *inferencial* (*fa*), medindo o poder de predição da rede *Bayesiana* sobre um conjunto de dados, de acordo com um critério de evidência (*Ie*). O programa foi executado em um cluster de memória compartilhada e os resultados mostraram que, para pequenas redes, a ordenação pode ser otimizada.

O algoritmo de estimação de distribuição (EDA – *Estimation of Distribution Algorithm*) [37] é um paradigma para computação evolucionária que tem sido usado como ferramenta de busca em problemas de aprendizado de estrutura em redes *Bayesianas*. Em [52], os autores usam duas versões de EDA, chamadas de UMDA e MIMIC, para obter a ordenação de variáveis para o algoritmo K2. Outros trabalhos que tratam da ordenação de variáveis para aprender redes *Bayesianas* a partir de dados e não usam abordagens de algoritmos evolucionários podem ser encontrados em [52].

4.4 Considerações Finais

Mesmo tendo vários trabalhos tratando da combinação de algoritmos evolucionários e redes *Bayesianas*, muitos deles não são definidos para aprender classificadores *Bayesianos* e sim redes *Bayesianas* irrestritas. Um método para induzir redes *Bayesianas* a partir de dados, projetada para a tarefa de classificação, é proposto em [44]; a tarefa de classificação,

entretanto, é considerada apenas para definir a função de aptidão, a qual é baseada na taxa de classificação.

O capítulo seguinte apresenta métodos híbridos, propostos neste trabalho, para o aprendizado de classificadores *Bayesianos* a partir de dados. Os métodos, chamados de VOGA, VOGAC e VOEA, aplicam algoritmos evolucionários na busca de uma ordenação de variáveis adequada para otimizar o processo de aprendizado. A descrição dos métodos e os resultados obtidos nos experimentos também são detalhados na seqüência. Além disso, é apresentado um novo algoritmo de aprendizado de classificadores *Bayesianos*, denominado de K2-MBC, para ser empregado em conjunto com os algoritmos evolucionários na busca da melhor ordenação de variáveis.

5 MÉTODOS E RESULTADOS

A ordenação prévia das variáveis utilizadas na definição de um problema é proposta, neste trabalho, como otimização do aprendizado de redes *Bayesianas,* voltadas para tarefa de classificação. Para isto, foram desenvolvidos métodos híbridos que empregam algoritmos evolucionários e algoritmos de aprendizado de redes *Bayesianas* para encontrar a melhor ordenação.

No capítulo anterior, foram apresentadas algumas abordagens que utilizam algoritmos evolucionários na busca da ordenação das variáveis. No entanto, estas abordagens focam principalmente na otimização do aprendizado de redes *Bayesianas* irrestritas. Já este trabalho, focaliza a otimização do aprendizado de redes *Bayesianas* para classificação.

Neste capítulo, são apresentados os métodos híbridos propostos e os resultados obtidos nos experimentos.

5.1 Descrição dos Métodos Híbridos

Os métodos híbridos, desenvolvidos neste projeto, aplicam a teoria de algoritmos evolucionários e aprendizado de redes *Bayesianas*, visando a otimização deste aprendizado, através da busca de uma ordenação adequada das variáveis. As redes *Bayesianas* induzidas pelos métodos híbridos são destinadas a tarefas de classificação, sendo necessária a escolha de uma variável para ser usada como classe durante todo o processo.

O processo de busca da ordenação das variáveis, realizado pelos métodos criados, se resume no seguinte: possíveis ordenações, geradas aleatoriamente, são codificadas como cromossomos, formando uma população inicial. Os cromossomos guardam a variável classe na primeira posição e são avaliados por uma função de aptidão, a qual é definida por

algoritmos de aprendizado de redes *Bayesianas*. Inicia-se um processo iterativo onde os cromossomos mais aptos são selecionados e submetidos aos operadores genéticos para formar uma nova geração. Este processo se repete até que a condição de parada seja alcançada (definida para os métodos propostos como o número máximo de gerações sem melhoria na população). Ao final do processo, tem-se o melhor cromossomo, contendo a melhor ordenação encontrada, e o classificador correspondente.

Os métodos híbridos propostos foram chamados de: VOGA (*Variable Ordering Genetic Algorithm*) [55], VOGAC (*Variable Ordering Genetic Algorithm using Classification*) [56] e VOEA (*Variable Ordering Evolutionary Algorithm*). VOGA e VOGAC são algoritmos genéticos que se diferenciam apenas na função de aptidão. VOEA é um algoritmo evolucionário que não utiliza o operador de mutação. As Figuras 5.1 e 5.2 apresentam os fluxogramas destes métodos.

Na Figura 5.1 a), nota-se que VOGA parte de um conjunto de treinamento (Tr), passado como entrada pelo usuário. Em seguida, é gerada uma população inicial (Pop_I, onde I corresponde a iteração), composta por N cromossomos que representam as possíveis Ordenações de Variáveis (OVs). Cada ordenação é fornecida como entrada para o algoritmo de aprendizado baseado em busca heurística para a indução da rede *Bayesiana*. O valor do *score Bayesiano,* calculado pelo algoritmo de aprendizado, é usado como aptidão para avaliar os cromossomos. No caso de se usar o algoritmo K2 como algoritmo de aprendizado, o valor de g (função (4), Seção 2.2.1) é usado como aptidão. Os cromossomos com os melhores valores de *score* terão maiores chances de serem escolhidos pelo método de seleção para a próxima geração. Após a seleção, são aplicados os operadores de cruzamento e mutação para gerar novos cromossomos. A população é novamente avaliada e os valores de g armazenados para cada cromossomo. Se o método obteve uma geração com pelo menos um cromossomo melhor que os cromossomos da geração anterior, haverá outra seleção para compor uma nova

população, repetindo todo processo. Caso contrário, são retornados a melhor ordenação encontrada e o classificador *Bayesiano* correspondente.

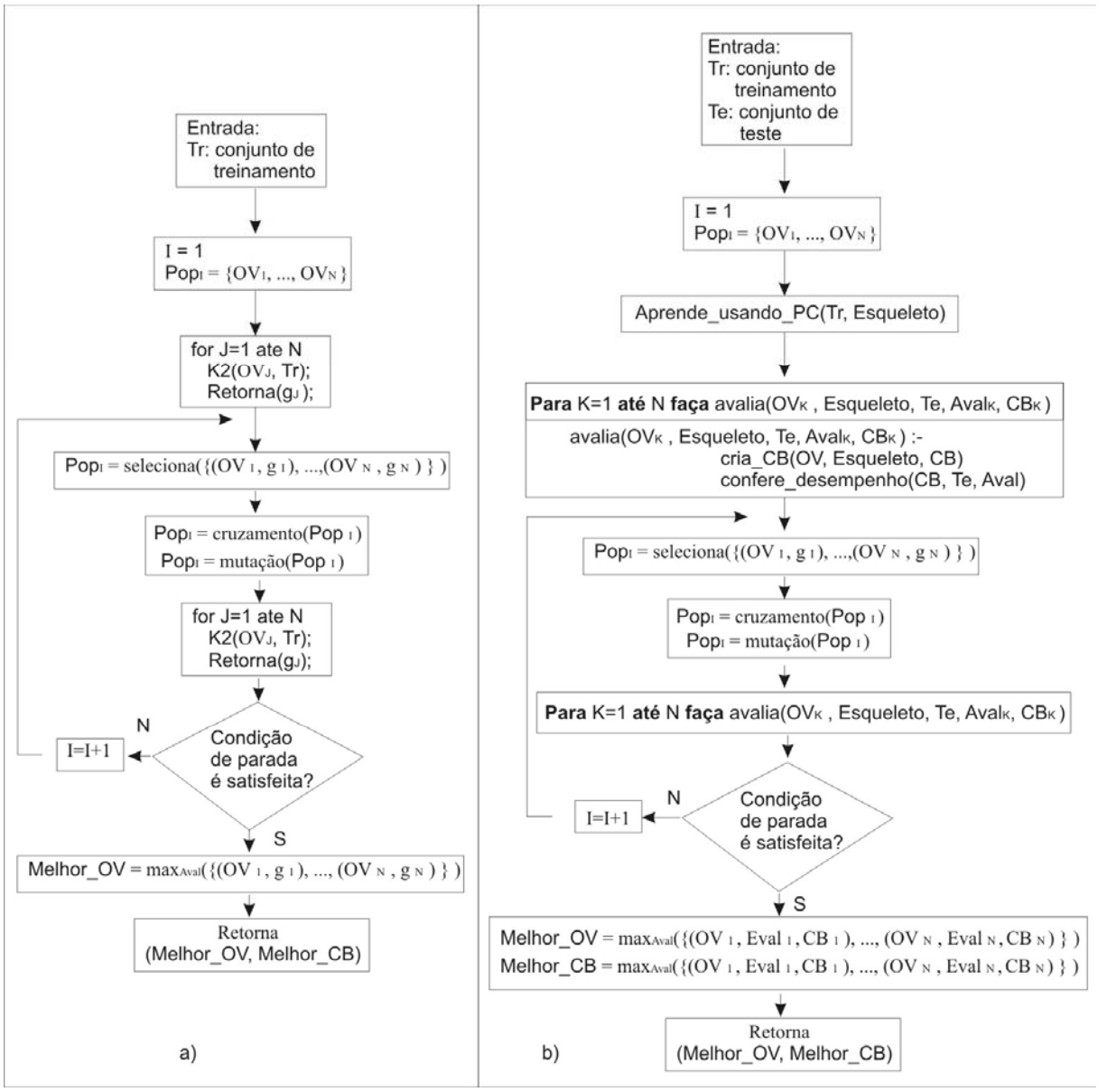


Figura 5.1. a) Fluxograma do VOGA e b) fluxograma do VOGAC.

A versão VOGAC, descrita na Figura 5.1 b), recebe um conjunto de treinamento e um conjunto de teste. Após a geração da população inicial (*Pop$_I$)*, o algoritmo PC é usado para gerar a estrutura não-direcionada (esqueleto) da rede *Bayesiana*. Os cromossomos, representando as ordenações, são usados para direcionar os arcos da estrutura. Cada ordenação cria, assim, uma nova rede *Bayesiana*, utilizada como classificador para classificar

o conjunto de teste. A taxa de classificação fornecida é usada como aptidão para avaliar cada cromossomo. Depois de avaliados, os cromossomos são selecionados e expostos aos operadores de cruzamento e mutação, gerando uma nova população. A nova população também é avaliada. A condição de parada é verificada e, se não for satisfeita, o algoritmo repete o processo a partir da seleção dos cromossomos. Caso contrário, são retornados a melhor ordenação encontrada e o classificador *Bayesiano* correspondente.

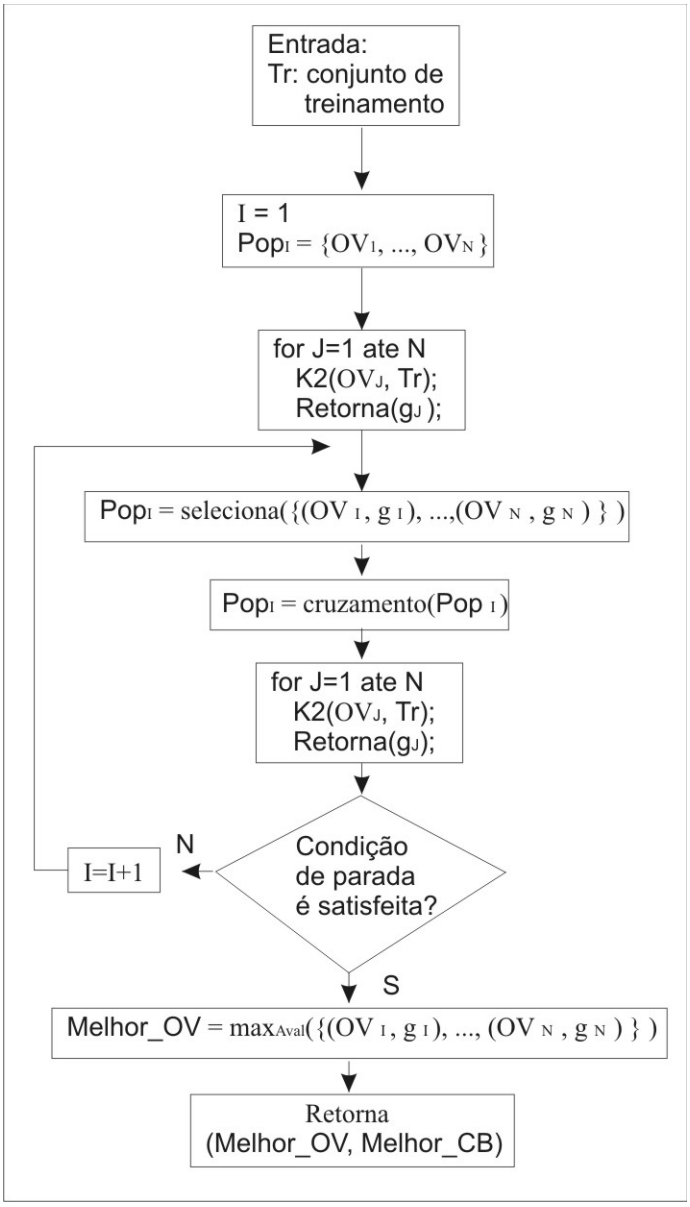


Figura 5.2. Fluxograma representando o VOEA.

O fluxograma da Figura 5.2 revela que VOEA é parecido com VOGA, porém não

aplica o operador de mutação. Desta forma, reduz o número de operações usadas na busca da ordenação adequada. Na Seção 5.2, são descritos os parâmetros genéticos escolhidos para os métodos.

5.2 Parâmetros Genéticos dos Métodos Híbridos Propostos

Inicialmente, os parâmetros genéticos dos métodos VOGA, VOGAC e VOEA foram definidos, de acordo com as especificações dadas na Seção 3.6. Posteriormente, alguns destes parâmetros serão alterados para a avaliação dos métodos. Veja, nas subseções a seguir, os parâmetros determinados.

5.2.1 Representação

As variáveis das bases de dados são representadas por números inteiros positivos. Cada número inteiro positivo é chamado de gene e são agrupados em ordenações diferentes para formarem os cromossomos da população. Desta forma, cada cromossomo codifica uma possível ordenação das variáveis. Como a ordenação adequada visa otimizar o aprendizado de classificadores *Bayesianos*, a primeira variável é considerada a variável classe e é mantida fixa na primeira posição do cromossomo durante todo o processo de busca. Desta forma, cada cromossomo tem (*n-1*) genes que podem ser trocados de posição, onde *n* é o número total de variáveis (incluindo a variável classe). Assim, qualquer permutação de *1* a *n-1* é um cromossomo em potencial. Por exemplo, em um domínio com 3 variáveis, *V1, V2* e *V3* (e uma classe), há 6 possíveis ordenações (possíveis cromossomos), como mostrado na Figura 5.3.

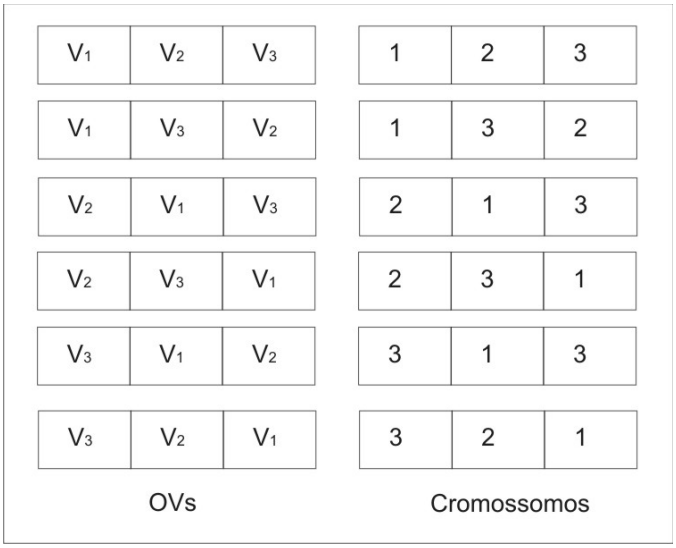


Figura 5.3. Possíveis ordenações de variáveis e suas representações como cromossomos para um domínio de 3 variáveis (V1, V2 e V3).

5.2.2 Definição da População Inicial

A população inicial de VOGA, VOGAC e VOEA é composta por cromossomos gerados aleatoriamente, a partir das variáveis da base de dados, mantendo-se a variável classe, escolhida pelo usuário, fixa durante todo o processo de busca na primeira posição de cada cromossomo.

O tamanho da população é calculado de acordo com o número de variáveis da base de dados. Considere $|OV|$ o número de Ordenações de Variáveis possíveis e $|V|$ o número de Variáveis do domínio, o tamanho da população inicial $|Pop_1|$ é dado pela função descrita em (8), a qual foi empiricamente construída.

```
se |V| < 5
   então tamanho (Pop1) = 6
   senão se (|OV|*0,05) > 100
            então
                 tamanho (Pop1) = 100
            senão
                 tamanho (Pop1) = ( |OV|*0,05 )
```

(8)

5.2.3 Função de Aptidão

Os métodos VOGA, VOGAC e VOEA utilizam algoritmos de aprendizado de redes *Bayesianas* para definir a função de aptidão. Os cromossomos da população, representando as ordenações das variáveis, servem como entrada para o algoritmo de aprendizado escolhido. Os classificadores gerados na saída dos algoritmos de aprendizado são avaliados pela função de aptidão, o que reflete quanto cada ordenação é adequada.

Os métodos VOGA e VOEA utilizam apenas algoritmos de aprendizado baseados em busca heurística para aprender os classificadores a partir da ordenação nos cromossomos. A função de aptidão destes métodos é definida pelo *score Bayesiano* dos classificadores *Bayesianos* aprendidos. Assim, cada cromossomo recebe como nota (aptidão) o valor deste *score.*

VOGAC pode utilizar algoritmos de aprendizado baseados em busca heurística, ou baseados no conceito de independência condicional para definir a aptidão dos cromossomos. A função de aptidão é dada pela taxa de classificação dos classificadores aprendidos.

5.2.4 Operadores Genéticos

a) **Operador de Seleção:** Depois de avaliados pela função de aptidão, os cromossomos da população são selecionados pelo método de seleção por torneio (Seção 3.2.2). Os cromossomos mais aptos (com maior nota) têm mais chances de serem selecionados. Os cromossomos que não forem selecionados são descartados.

b) **Operador de Cruzamento:** Nos cromossomos selecionados pelo método de seleção, é aplicado o operador de cruzamento OX (Seção 3.4). A taxa de cruzamento utilizada

nos experimentos iniciais foi de 0.8. O método VOEA encerra a formação da nova geração nesta etapa do processo, pois não utiliza o operador de mutação.

c) **Operador de Mutação:** A formação da nova geração, em VOGA e VOGAC, é finalizada com aplicação do operador de mutação baseado na posição (Seção 3.4). A taxa de mutação, fornecida nos experimentos inicias, foi de 30% dos genes dos cromossomos, calculada de acordo a função (9):

$$Tm = n*c*0.3, \quad (9)$$

onde *n* é o número de variáveis da base de dados e *c* é o número de cromossomos da população. Na tentativa de minimizar o esforço computacional exigido em cada iteração do algoritmo, o método VOEA não aplica o operador de mutação nos cromossomos.

d) **Critério de Parada:** O critério de parada, definido para os métodos híbridos propostos, refere-se ao número máximo de gerações sem melhoria na população. Inicialmente, é fixado um número máximo de gerações, no qual os cromossomos da nova população não apresentem aptidões melhores que os cromossomos das populações anteriores. Se, após este número máximo de gerações, não surgir pelo menos um cromossomo que tenha o valor de aptidão melhor que todos os cromossomos das populações anteriores, o método de busca é finalizado e retorna a melhor ordenação encontrada e o classificador *Bayesiano* correspondente. Nos experimentos iniciais, foi fixado empiricamente para o critério de parada, no máximo, 10 gerações sem melhoria.

5.3 Descrição das Bases de Dados

Para realizar os experimentos, foram utilizados oito domínios, sendo: i) três domínios de redes *Bayesianas* conhecidos na literatura: *Alarm* [18], *Asia* [18] e *Engine* [18]; ii) três problemas de *benchmark* extraídos do repositório da Universidade da Califórnia, em *Irvine* [39], chamados *Balance, Iris,* e *Congressional Voting Records (Voting)*; iii) e dois domínios, *Synthetic 1* e *Synthetic 2,* foram criados especificamente para os experimentos.

A principal motivação de usar os domínios *Asia, Engine, Synthetic 1* e *Synthetic 2* é o fato de suas redes *Bayesianas* já serem conhecidas e, assim, é possível identificar uma ordenação de variáveis a *priori* para cada domínio. Dessa forma, os resultados dos métodos podem ser comparados com os casos ótimos.

Os domínios artificiais (*Synthetic 1* e *Synthetic 2*) foram criados no software *GENIE* [66] usando uma estratégia de amostragem aplicada aos classificadores *Bayesianos* mostrados na Figura 5.4. *Synthetic 1* descreve um domínio, onde apenas 1 variável influencia diretamente a variável classe. *Synthetic 2*, contudo, descreve um domínio, no qual 14 variáveis influenciam diretamente a variável classe.

A base de dados *Alarm* foi utilizada apenas nos experimentos da Subseção 5.4.1, devido ao grande número de atributos e à falta de tempo para execução com os métodos híbridos. A Tabela 5.1 resume as características das bases de dados usadas.

Tabela 5.1. Descrição das bases de dados com nome das bases, número de atributos mais a classe (AT), número de instâncias (IN) e número de valores da classes (CI).

	Asia	Alarm	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
AT	8	37	5	9	5	32	32	17
IN	15000	5000	625	15000	150	5000	5000	232
CI	2	2	3	2	3	2	2	2

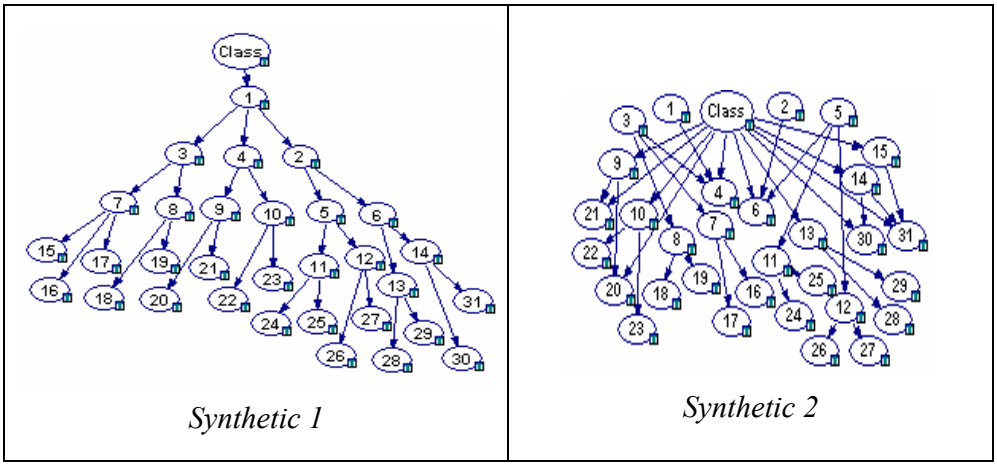


Figura 5.4. Redes Bayesianas representando os domínios Synthetic 1 e Synthetic 2. As representações gráficas foram criadas usando o software Genie [66].

5.4 Algoritmos do método VOGA

O método VOGA emprega algoritmos de aprendizado de redes *Bayesianas* baseados em busca heurística para induzir a estrutura dos classificadores *Bayesianos*. Estes algoritmos de aprendizado são usados na definição da função de aptidão. A flexibilidade da escolha do algoritmo de aprendizado permitiu desenvolver, neste trabalho, dois algoritmos variantes de VOGA, chamadas de: VOGA-K2 e VOGA-K2-MBC.

Os dois algoritmos desenvolvidos, VOGA-K2 e VOGA-K2-MBC, esperam, como entrada, um conjunto de treinamento e retornam a melhor ordenação de variáveis encontrada e o classificador correspondente, como pode ser visto no fluxograma da Figura 5.1 a).

A função de aptidão de VOGA-K2 e VOGA-K2-MBC é definida segundo o *score Bayesiano*, calculado pelo algoritmo de aprendizado durante a indução da rede *Bayesiana*. VOGA-K2 utiliza o algoritmo de aprendizado K2. O *score Bayesiano*, calculado pelo K2, é dado pela função *g* (veja função (4), Seção 2.2.1), sendo usado por VOGA-K2 como valor de aptidão para os cromossomos da população. Já o VOGA-K2-MBC utiliza um novo algoritmo de aprendizado de classificadores *Bayesianos*, chamado K2-MBC. Este novo algoritmo está sendo proposto neste trabalho e também utiliza a função *g* no aprendizado da estrutura do

classificador, fornecendo assim a aptidão para os cromossomos.

5.4.1 Classificador K2-MBC

O algoritmo K2-MBC (*K2-Based Markov Blanket Classifier*) é uma versão do algoritmo K2 e explora o conceito de *Markov Blanket* (apresentado no capítulo 2) para minimizar o esforço computacional durante o aprendizado de classificadores *Bayesianos.* O algoritmo K2 foi definido para aprender redes *Bayesianas* irrestritas. Por outro lado, o K2-MBC deve ser usado apenas para problemas de classificação.

A idéia principal do algoritmo é estender o *Markov Blanket* da variável classe (CMB) e excluir do conjunto de possíveis estruturas aquelas que têm atributos fora do CMB. A Figura 5.5 descreve o algoritmo K2 e o algoritmo K2-MBC.

Analisando a Figura 5.5, percebe-se que o K2-MBC diferencia-se do K2 apenas nas linhas 11, 14 e 15. As linhas 10, do K2, e 11, do K2-MBC, são similares. No entanto, no K2-MBC, a linha 11 testa se o atributo *z* não está presente no *Markov Blanket* da variável classe (CMB). Então, os atributos identificados como membros do CMB não são testados como candidatos a serem pais e não são incluídos como pais de outros atributos. Assim, o número de estruturas possíveis a serem testadas é reduzido.

As linhas 14 e 15 do K2-MBC não estão presentes no K2. Estas linhas são responsáveis pela identificação do conjunto de atributos que formam o CMB. Vale lembrar que o K2-MBC utiliza-se de uma heurística para definir o CMB e, desta forma, o conjunto de atributos identificados como pertencentes ao CMB é um conjunto aproximado.

O algoritmo K2-MBC foi implementado em duas versões diferentes: i) permitindo no máximo dois pais para cada atributo e ii) permitindo no máximo 10 pais para cada atributo. A idéia de limitar o número de atributos pais é verificar a diferença no tempo de construção das

redes e na exatidão da classificação.

```
1. Algoritmo K2:

2. {Entrada: Um conjunto de n nós, uma ordenação dos nós, limite superior u sobre o
3.            número de pais que um nó pode ter e uma base de dados D, contendo m casos.}
4. {Saída: Para cada nó, a identificação dos pais do nó.}
5.    for i := 1 to n do
6.        πi := {};
7.        Pold := g(i, πi); {Esta função é calculada usando a equação (4).}
8.        OKToProceed := true
9.        while OKToProceed and | πi | < u do
10           suponha z ser o nó em Pred(xi) - πi que maximiza g(i, πi, ∪ {z});
11.          Pnew := g(i, πi, ∪ {z});
12.          if Pnew > Pold then
13.              Pold := Pnew;
14.              πi := πi ∪ {z};
15.          else OKToProceed := false;
16.     end {while};
17.     write ('Nó:', xi, 'Pais deste nó:',  πi)
18. end {for};
19. end {K2};
```

```
1. Algoritmo K2-MBC:

2.   {Entrada: Um conjunto n de nós, uma ordenação dos nós, um limite superior sobre
3.           o número de pais que um nó pode ter e uma base de dados D, contendo m casos.}
4.   {Saída: Para cada nó, uma identificação dos pais desse nó.}
5.   CMB := { };   /* Class Markov Blanket */
6.   for i := 1 to n do
7.        πi := { };
8.        Pold := g(i, πi); {Esta função é calculada usando a equação (4).}
9.        OKToProceed := true
10.       while OKToProceed and | πi | < u do
11  .         Suponha z ser o nó em Pred(xi) – {πi ∪ MB}, que maximiza g(i, πi, ∪ {z});
12.           Pnew := g(i, πi, ∪ {z});
13.           if Pnew > Pold then
14.                   if (z == classe) or (z é pai de qualquer filho de classe) então
15                        MB := MB ∪ {z};
16.               Pold := Pnew;
17.               πi := πi ∪ {z};
18.           else OKToProceed := false;
19.       end  /*while*/;
20.       escreve ('Nó:', xi, 'Pais deste nó:',  πi)
21.  end  /*for*/
22. end /* K2-MBC*/
```

Figura 5.5. Algoritmo K2-MBC e algoritmo K2.

Os experimentos realizados com K2-MBC utilizaram as bases de dados descritas na Seção 5.3. As bases de dados da Tabela 5.1 foram divididas em 10 conjuntos de treinamento e testes. Os conjuntos de treinamento, em conjunto com a ordenação original das variáveis

encontrada nas bases de dados, foram usados para o aprendizado dos classificadores. Os conjuntos de testes foram usados para realizar a classificação (num processo de validação-cruzada com 10 pastas) e assim avaliar a qualidade dos classificadores gerados. A primeira variável de cada base de dados foi escolhida como variável classe.

As Tabelas 5.2 e 5.3 mostram os resultados da Média da Taxa de Classificação Correta (MTCC), obtidos por K2-MBC, K2 e *Naive Bayes*. Também apresentam o número de vezes que a função g foi calculada (Nro. de g) por K2-MBC e K2, demonstrando o custo computacional destes algoritmos.

A Tabela 5.2 revela que, quando se considera a taxa de classificação correta, o algoritmo K2-MBC consegue resultados tão bons quanto K2 e ambos (K2 e K2-MBC) obtiveram um desempenho superior ao *Naive Bayes* nas bases *Alarm* e *Synthetic* 2. Na Tabela 5.3, os quatro métodos conseguiram resultados muito similares para classificação, sendo difícil dizer qual se saiu melhor, com exceção de um ganho maior de K2 na base *Voting*.

Tabela 5.2. Média da Taxa de Classificação Correta (MTCC) obtida por K2-MBC, K2 e Naive Bayes e número de vezes que a função g foi calculada (Nro. de g) por K2-MBC e K2.

	Alarm		**Asia**		**Engine**		**Synthetic 2**	
	MTCC	**Nro. de g**	**MTCC**	**Nro. de g**	**MTCC**	**Nro. de g**	**MTCC**	**Nro. de g**
K2	96.93	2225.7	51.78	63	**99.60**	85	**93.35**	1059.8
K2-MBC 2	**96.98**	**140**	51.95	**19**	**99.60**	**28**	**93.35**	**154**
K2-MBC 10	96.96	183	**52.45**	21	**99.60**	33	**93.35**	179
Naive Bayes	50.32	-	51.78	**-**	99.52	-	54.70	-

Tabela 5.3. Média da Taxa de Classificação Correta (MTCC) obtida por K2-MBC, K2 e Naive Bayes e número de vezes que a função g foi calculada (Nro. de g) por K2-MBC e K2.

	Iris		**Synthetic 1**		**Balance**		**Voting**	
	MTCC	**Nro. de g**	**MTCC**	**Nro. de g**	**MTCC**	**Nro. de g**	**MTCC**	**Nro. de g**
K2	89.40	21	**89.35**	987.5	61.90	21	**96.75**	447.3
K2-MBC 2	89.40	**9**	84.30	**82.9**	61.90	**9.3**	92.23	**52**
K2-MBC 10	89.40	**9**	84.30	**82.9**	61.90	**9.3**	92.23	54.4
Naive Bayes	**90.00**	**-**	84.44	**-**	**62.28**	-	92.23	-

A restrição no número de pais degradou os resultados da classificação nas bases *Alarm* e *Asia*, mas nas outras bases de dados, limitar o número de pais de cada atributo não

influenciou na MTCC.

Quanto ao custo computacional, *Naive Bayes* é imbatível comparado a K2 e K2-MBC, porém os resultados apresentados mostram que K2-MBC pode se sair melhor, em algumas bases, que *Naive Bayes* para a classificação e com um custo computacional bem menor que K2, quando este é usado para aprender classificadores *Bayesianos.*

O algoritmo K2-MBC foi desenvolvido com o objetivo de melhorar o tempo computacional do VOGA, sem comprometer a qualidade dos classificadores gerados. Os resultados dos experimentos realizados com os algoritmos de VOGA-K2 e VOGA-K2-MBC são apresentados na subseção seguinte.

5.4.2 Resultados obtidos com o método VOGA

Os experimentos realizados com VOGA-K2 e VOGA-K2-MBC envolveram os parâmetros genéticos definidos na Seção 5.2 e as bases de dados descritas na Seção 5.3. Os passos abaixo descrevem como esses experimentos foram conduzidos.

1 Inicialmente, usando uma estratégia de validação cruzada (10 *partes*), foram gerados 10 conjuntos de treinamento e 10 conjuntos de testes a partir de cada base de dados. A variável classe foi definida para cada base de dados.

2 Os conjuntos de treinamento foram usados como entrada para os algoritmo K2 e VOGA-K2. O algoritmo K2 utilizou, como ordenação inicial, a ordem em que as variáveis aparecem nas bases de dados.

3 Os mesmos conjuntos de treinamento, usados no passo 2, foram usados como entrada para o K2-MBC e VOGA-K2-MBC. K2-MBC utilizou, como ordenação inicial, a ordem em que as variáveis aparecem nas bases de dados.

4 O *score Bayesiano* (g) obtido para cada conjunto de treinamento e o número de

gerações necessárias para alcançar a solução, nos passos 2 e 3, foram armazenados e as médias calculadas. O número de vezes em que a função g foi calculada por VOGA-K2 e VOGA-K2-MBC também foi armazenado para avaliação do desempenho dos dois algoritmos. Os resultados obtidos são apresentados na Tabela 5.4 e nas Figuras 5.7 e 5.8.

A idéia original do VOGA é utilizar um algoritmo genético para encontrar uma ordenação adequada de variáveis para melhorar os resultados do algoritmo K2 e K2-MBC. Assim o *score* (função *g*) do classificador obtido, usando VOGA-K2 e VOGA-K2-MBC, deveria ser melhor que o (ou, no mínimo, igual ao) *score* obtido quando for usado apenas os algoritmo K2 e K2-MBC. Então, os experimentos foram projetados, tentando observar os efeitos de VOGA-K2 e VOGA-K2-MBC no *score* do classificador. Em outras palavras, os experimentos executados geraram resultados que permitem uma análise comparativa entre VOGA-K2 e K2 e VOGA-K2-MBC e K2-MBC, em termos da função *g*.

É importante ressaltar que para as bases de dados *Asia, Engine, Synthetic 1* e *Synthetic 2* a estrutura original da rede Bayesiana é conhecida e, desta forma, é possível definir ordenações adequadas para o aprendizado, isto foi feito para as execuções do K2 e do K2-MBC. Uma ordenação adequada é aquela que não impede nenhum arco presente na rede original. Considere por exemplo a rede *Bayesiana R1* dada na Figura 5.6, com as variáveis V1, V2, V3 e V4. Uma ordenação V1, V2, V3 e V4 é uma ordenação adequada para esta rede, pois as variáveis pais aparecem sempre antes de suas respectivas filhas. Já a ordenação V4, V3, V2 e V1 não é adequada.

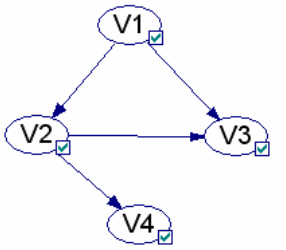


Figura 5.6. Estrutura da rede *Bayesiana R1*.

Pelo fato dos algoritmos K2 e K2-MBC serem executados tendo como entrada ordenações de variáveis adequadas para as 4 bases citadas anteriormente, seus desempenhos para estas bases podem ser considerados próximos do desempenho ótimo. Assim, se os resultados obtidos com o VOGA estiverem próximos dos obtidos com o K2 e K2-MBC, nestas bases específicas, podem ser considerados bons.

Tabela 5.4. Média dos *scores Bayesianos* (função g) obtidos pelo K2, VOGA-K2, K2-MBC e VOGA-K2-MBC.

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
K2	-39742,44	-2254,79	-30386,81	-1835,88	**-76318,1**	-79065,62	-1569,34
VOGA-K2	**-39741,37**	-2254,79	**-30386,79**	-1835,88	-76351,22	**-79061,72**	**-1548,83**
K2-MBC	-46920,53	-2254,79	-30456,43	-1835,88	-88902,01	-88961,21	-1810,87
VOGA-K2-MBC	**-46502,08**	-2254,79	-30456,43	-1835,88	**-86363,7**	-88961,21	**-1809,37**

Analisando os resultados apresentados na Tabela 5.4, é possível deduzir que o algoritmo VOGA-K2 produziu resultados, no mínimo, tão bons quanto o K2. Em 4 das 7 bases de dados, os resultados obtidos por VOGA-K2 foram melhores que os obtidos por K2. Um outro fato interessante, revelado pela Tabela 5.4, é que as bases de dados, com maior número de atributos, favoreceram o método VOGA-K2. Considerando-se os algoritmos K2-MBC e VOGA-K2-MBC, os classificadores *Bayesianos* aprendidos com VOGA-K2-MBC possuem *score Bayesianos* tão bons quanto os aprendidos por K2-MBC, sendo que, em três domínios (*Asia*, *Synthetic 1* e *Voting),* VOGA-K2-MBC produziu resultados melhores.

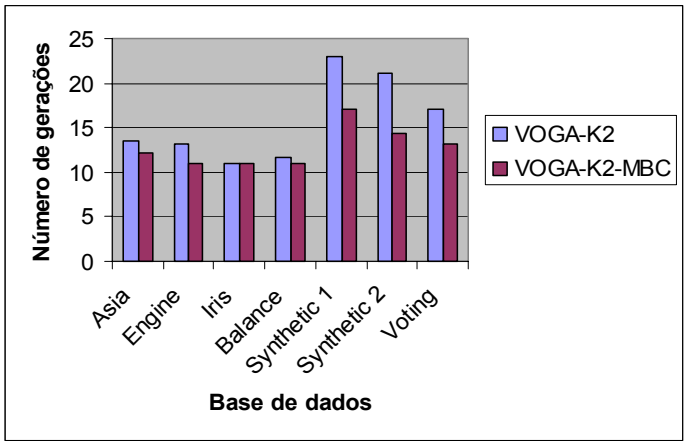


Figura 5.7. Número de gerações necessárias até a convergência.

Os resultados apresentados na Tabela 5.4 mostram que VOGA-K2 e VOGA-K2-MBC são capazes de encontrar boas ordenações, resultando em classificadores com *score Bayesianos* melhores que os aprendidos por K2 e K2-MBC. Vale ressaltar que o custo computacional de VOGA-K2-MBC é bem menor que de VOGA-K2, como pode ser observado nas Figuras 5.7 e 5.8, que comparam número de gerações necessárias para a convergência e o número de vezes em que a função g foi executada pelos dois algoritmos, respectivamente.

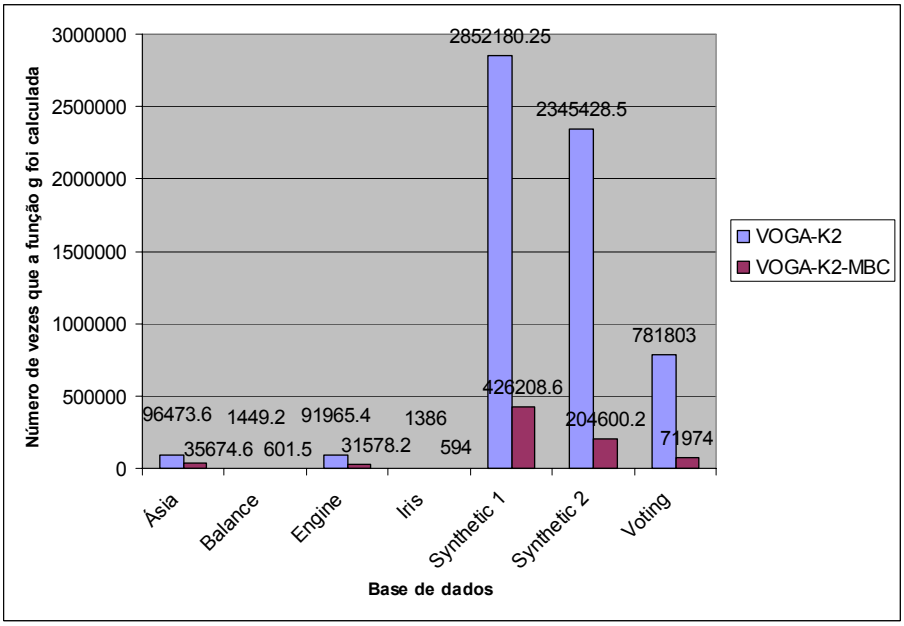


Figura 5.8. Média do número de vezes que a função g foi calculada por VOGA-K2 e VOGA-K2-MBC.

5.5 Algoritmos do método VOGAC

O método VOGAC diferencia-se de VOGA na definição da função de aptidão. A idéia do VOGAC é utilizar a Taxa de Classificação Correta (TCC) para avaliar as ordenações de variáveis codificadas nos cromossomos. Deste modo, não depende dos *scores Bayesianos* como VOGA e, por isso, permite o uso de algoritmos de aprendizado, baseados em busca heurística, ou baseados em conceitos de independência condicional (os quais não geram *score Bayesiano*), para gerar as estruturas dos classificadores *Bayesianos*.

Teoricamente, a vantagem de utilizar a TCC como função de aptidão possibilita que se utilize um algoritmo de aprendizado baseado em Independência Condicional uma única vez e, assim, o VOGAC pode ser mais rápido que VOGA, pois a busca da melhor ordem não precisa executar algoritmos de aprendizado (que são mais complexos) várias vezes, como no VOGA. Ao invés disso, executa-se várias vezes a classificação, a qual é linear em relação ao tamanho da base de dados de teste.

Como pode ser visto no fluxograma da Figura 5.1 b), o algoritmo de aprendizado é executado uma única vez para gerar a estrutura do classificador. De posse desta estrutura, VOGAC utiliza a ordenação, representada em cada cromossomo, para direcionar os arcos, conforme explicado na Seção 2.4. Assim, os cromossomos podem originar classificadores *Bayesianos* diferentes e serem avaliados, conforme a TCC, ao seguir as etapas de busca – geração da população inicial, seleção por torneio, *crossover* OX e mutação baseada na troca de posições – até não mais surgir cromossomos melhores que os das gerações anteriores.

O algoritmo de aprendizado utilizado pelo VOGAC, nos experimentos deste trabalho, foi o PC, gerando o algoritmo chamado VOGAC-PC.

O algoritmo VOGAC-PC foi desenvolvido para induzir classificadores *Bayesianos* a partir de dados, utilizando o algoritmo de aprendizado baseado em independência condicional,

conhecido como PC (Seção 2.3.4). O VOGAC-PC espera como entrada um conjunto de treinamento (Tr) e um conjunto de testes (Te). Após a conclusão do processo de busca, são retornados a melhor ordenação de variáveis encontrada e o classificador *Bayesiano* correspondente. Os resultados obtidos com VOGAC-PC são apresentados na subseção seguinte.

5.5.1 Resultados obtidos com o método VOGAC

Os parâmetros genéticos, utilizados nestes experimentos, foram definidos na Seção 5.2 e as bases de dados utilizadas foram descritas na Seção 5.3. Os experimentos foram conduzidos de acordo com os seguintes passos:

1. Os conjuntos de treinamento e testes, usados para a execução das abordagens de VOGA, foram usados para a execução das abordagens de VOGAC. Os algoritmos K2 e PC também foram executados com os conjuntos de treinamento, usando a ordenação original das variáveis de cada conjunto de dados. A Média da Taxa de Classificação Correta (MTCC) foi armazenada.
2. O número de gerações necessárias para que o VOGAC-PC alcançasse a solução foi armazenado e comparado com as versões de VOGA.

Os resultados obtidos pelos algoritmos nos passos 1 e 2 são mostrados na Tabela 5.5 e na Figura 5.9, respectivamente.

Tabela 5.5. Média da Taxa de Classificação Correta Média (MTCC) .

	Asia	**Balance**	**Engine**	**Iris**	**Synthetic 1**	**Synthetic 2**	**Voting**
K2	51.78	61.90	99.59	89.33	89.35	93.35	96.75
PC	51.78	62.28	99.57	92.66	82.70	48.55	95.50
VOGA-K2	51.74	61.90	99.59	89.33	89.30	93.35	95.47
VOGA-K2-MBC	**52.00**	61.90	99.59	89.33	84.15	93.35	92.65
VOGAC-PC	51.84	**86.30**	**99.60**	**95.33**	89.15	**93.43**	96.71

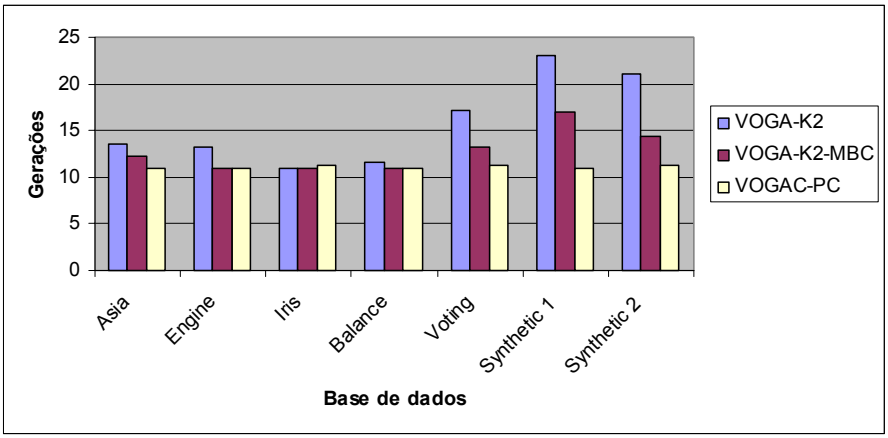


Figura 5.9. Número de gerações necessárias até a convergência.

Analisando os resultados da Tabela 5.5, é possível inferir que o VOGAC-PC obteve os melhores resultados de classificação em seis das sete bases de dados. É interessante notar, contudo, que, em alguns domínios (*Asia, Engine*), todos os algoritmos têm valores de MTCC muito próximas, o que torna difícil identificar a melhor.

Analisando a Figura 5.9, observa-se que o VOGAC-PC reduziu o esforço necessário para encontrar as melhores ordenações de variáveis, comparadas com as versões de VOGA.

Adicionalmente, a Tabela 5.5 revela que, nos domínios *Iris* e *Balance*, VOGAC-PC executou sensivelmente melhor que nos outros algoritmos. No domínio *Balance*, essa diferença é maior. No domínio *Iris*, é possível observar que os algoritmos que utilizaram PC foram melhores que os algoritmos que utilizaram K2. Para entender melhor esse comportamento, é interessante examinar a estrutura de rede descrita na Figura 5.10.

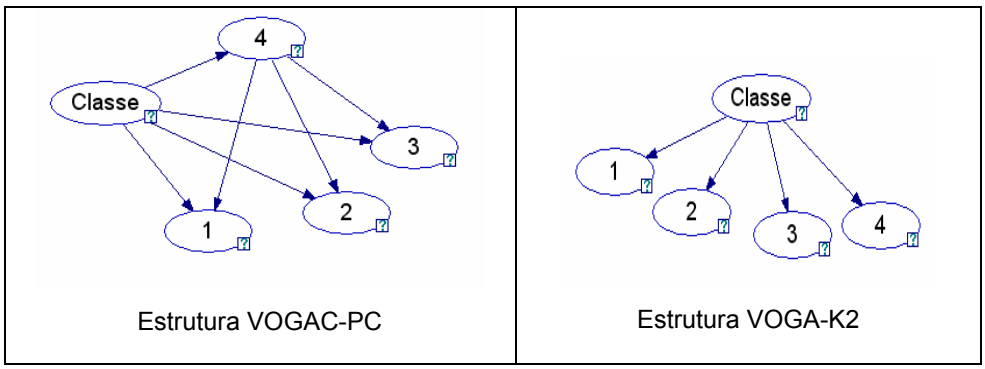


Figura 5.10. Estrutura dos classsificadores *Bayesianos* induzidos por VOGAC-PC e VOGA-K2, usando o domínio Iris. As representações gráficas foram criadas no software *Genie* [66].

Observando a Figura 5.10, fica claro que as estruturas encontradas por VOGAC-PC e VOGA-K2 são diferentes. Baseado neste fato, é possível concluir que o algoritmo PC executou melhor que o K2 nessa base de dados. O esqueleto definido pelo algoritmo PC provavelmente teve uma grande influência nos resultados.

5.6 Algoritmos do método VOEA

O método VOEA estende a idéia presente no VOGA, mas, ao invés de usar um algoritmo genético para encontrar uma ordenação de variáveis adequada (como em VOGA), aplica um algoritmo evolucionário para ajudar no processo de indução de um classificador *Bayesiano*. Este algoritmo evolucionário não usa o operador de mutação e, neste sentido, reduz o número de operações a serem efetuadas em cada geração, tentando assim diminuir o esforço computacional exigido pelo método.

A Figura 5.2 apresentou o algoritmo VOEA descrito como um fluxograma. É possível notar que VOEA espera como entrada um conjunto de instâncias para ser usado como um conjunto de treinamento e retorna a melhor ordenação e o classificador correspondente. As etapas de busca do algoritmo evolucionário são repetidas e, para cada geração, a melhor ordenação é guardada e passada para a seguinte. Se não houver melhoria depois de um número máximo de gerações, o algoritmo encerra e retorna a melhor ordenação encontrada, assim como o classificador correspondente aprendido.

O algoritmo de aprendizado utilizado por VOEA, nos experimentos deste trabalho, foi o K2, gerando o algoritmo chamado VOEA–K2. Os experimentos descritos na subseção seguinte mostram que bons resultados podem ser alcançados usando esta heurística.

5.6.1 Resultados obtidos com o método VOEA

Para os experimentos com o método VOEA, foram utilizados os parâmetros genéticos definidos na Seção 5.2 e as bases de dados descritas na Seção 5.3, sendo conduzidos conforme os passos abaixo:

1. Os conjuntos de treinamento e testes, usados para a execução das abordagens de VOGA e VOGAC, foram usados para a execução da abordagem de VOEA. Os valores do *scores Bayesianos* (função g) obtidos por VOEA, VOGA e K2 foram armazenados.
2. O número de gerações necessárias para que as abordagens de VOGA e VOEA alcançassem a solução foi armazenado.

Assim como o método VOGA, o método VOEA busca obter um classificador com *score* (função *g*) melhor que o *score* obtido quando é usado apenas o algoritmo K2 sem se conhecer uma ordenação adequada das variáveis. Então, os experimentos foram projetados, tentando observar os efeitos de VOEA no *score* do classificador. Desta forma, os experimentos executados geraram resultados que permitem uma análise comparativa entre VOEA e K2, em termos da função *g*. Além disso, pensando em fazer uma análise comparativa mais robusta, esta Seção mostra também os resultados obtidos com VOGA. Assim, é possível verificar se a eliminação do operador de mutação melhorou a velocidade de convergência.

A Tabela 5.6 apresenta os resultados dos *scores Bayesianos* obtidos por K2, VOEA-K2 e VOGA-K2. Pode-se ver que os *scores* obtidos pelo VOEA-K2 são melhores (ou no mínimo iguais) que os obtidos por VOGA-K2, exceto para três domínios (*Asia, Synthetic 1* e *Voting)*. K2 só é melhor que VOEA-K2 e VOGA-K2 para o domínio *Synthetic 1*, lembrando que para sua execução, já é passada a ordenação mais adequada.

Quanto à convergência, a Figura 5.11 mostra que VOEA-K2 alcança a solução final

mais rápido que VOGA-K2 na maioria dos domínios. Além disto, VOEA-K2 apresenta esforço computacional menor ou igual que VOGA-K2 em 4 dos 7 domínios, conforme o número de vezes que a função g foi calculada, como pode ser visto na Figura 5.12. Além de se considerar o número de gerações e o número de execuções da função g, é importante lembrar que o VOEA-K2 não realiza a operação de mutação (a qual é realizada em 30% dos genes de cada geração no VOGA-K2).

Tabela 5.6. Média do *scores Bayesianos* (função g) obtidos por K2, VOEA e VOGA-K2.

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
K2	-39742.44	-2254.79	-30386.81	-1835.88	**-76318.1**	-79065.62	-1569.34
VOEA-K2	-39742.46	-2254.79	**-30386.79**	-1835.88	-76352.52	**-79060.43**	-1549.32
VOGA-K2	**-39741.37**	-2254.79	**-30386.79**	-1835.88	-76351.22	-79061.72	**-1548.83**

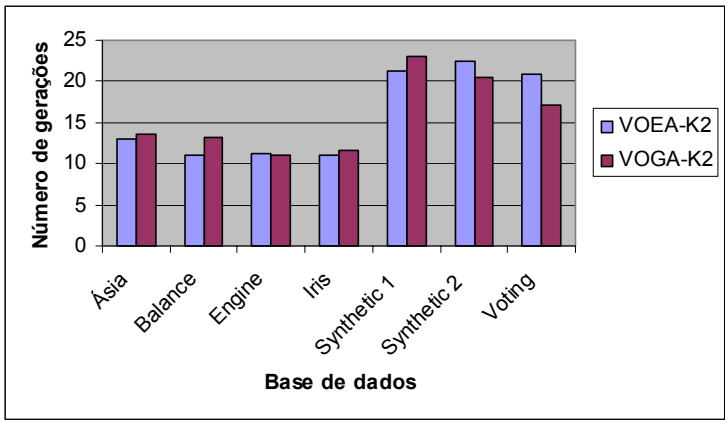


Figura 5.11. Número de gerações necessárias até a convergência.

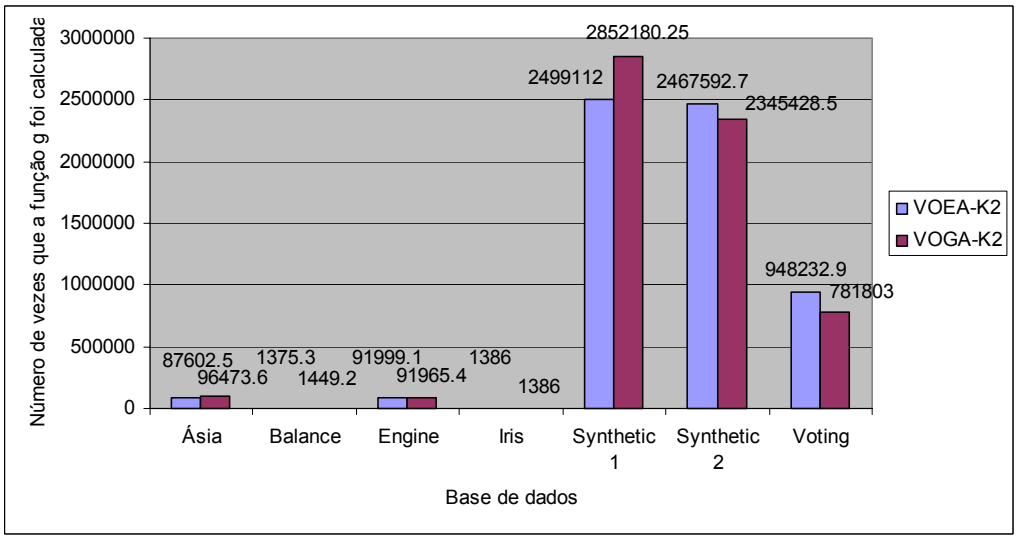


Figura 5.12. Número de vezes em que a função g foi executada por VOEA-K2 e VOGA-K2.

5.7 Alterações dos Parâmetros Genéticos

Como informado na Seção 3.6, a definição dos parâmetros genéticos pode influenciar no desempenho dos algoritmos evolucionários. Por isso, algumas modificações foram feitas nos parâmetros iniciais aplicados aos métodos VOGA, VOGAC e VOEA para verificar se os parâmetros escolhidos eram adequados. As subseções seguintes apresentam estas alterações.

5.7.1 Definição da População Inicial

Adicionalmente às abordagens de VOGA, VOGAC e VOEA, foram desenvolvidas abordagens um pouco diferentes, chamadas de VOGA+, VOGAC+ e VOEA+. Nestas novas abordagens, a população inicial não é gerada aleatoriamente e são usadas mais informações sobre a variável classe. Para isto, foram utilizados testes estatísticos.

Em VOGA+ e VOGAC+, foi usado um teste estatístico para definir a ordenação das variáveis nos cromossomos da população inicial. O teste estatístico aplicado foi o *qui-quadrado* (χ^2 - *chi-square test*) [67]. Desta forma, a população inicial passou a ser formada por cromossomos que representam as ordenações de variáveis estabelecidas pelo teste estatístico χ^2.

Em VOEA+, foram usados mais 3 testes estatísticos, além do χ^2, para aumentar a diversidade dos cromossomos, visto que este algoritmo não aplica o operador de mutação. São eles: *Gain Ratio* [43], *Information Gain* [43] e *SymmetricalUncertAttributeEval* [67]. A população inicial agora é composta de cromossomos contendo cada uma das ordenações geradas por estes métodos, além de cromossomos com ordenações geradas aleatoriamente. A cada 10 cromossomos, 4 possuem as ordenações geradas pelos testes estatísticos e os outros 6 são gerados aleatoriamente.

Os testes estatísticos são executados usando cada variável juntamente com a variável classe. Assim, a força das relações de dependências entre cada variável e a classe pode ser medida. Obviamente, a relação entre os testes estatísticos e a melhor ordenação de variáveis não pode ser totalmente confiável, mas trabalhos anteriores [5], assim como os resultados apresentados nas Tabelas 5.7, 5.8, abaixo, mostraram que bons resultados podem ser alcançados usando essa heurística.

Tabela 5.7. Valor dos *scores Bayesianos* (função g) dos classificadores induzidos pelos algoritmos de VOGA, VOGA+, VOEA e VOEA+.

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
VOGA-K2	-39741.37	-2254.79	-30386.79	-1835.88	-76351.22	**-79061.72**	-1548.83
VOGA-K2+	**-39740.77**	-2254.79	-30386.79	-1835.88	**-76317.66**	-79061.84	**-1545.32**
VOGA-K2-MBC	-46502.08	-2254.79	-30456.43	-1835.88	**-86363.7**	-88961.21	-1809.38
VOGA-K2-MBC+	-46502.08	-2254.79	-30456.43	-1835.88	-86364.97	-88961.21	-1809.38
VOEA-K2	**-39742.46**	-2254.79	**-30386.79**	-1835.88	-76352.52	**-79060.43**	-1549.32
VOEA-K2+	-39744.13	-2254.79	-30439.66	-1835.88	**-76324.35**	-79061.4	**-1546.98**

Tabela 5.8. Média da Taxa de Classificação Correta (MTCC) obtida pelos algoritmos de VOGAC-PC e VOGAC-PC+.

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
VOGAC-PC	51.84	86.30	99.60	95.33	89.15	**93.43**	96.71
VOGAC-PC+	51.84	86.30	99.60	95.33	89.15	93.40	96.71

A Tabela 5.7 compara os *scores* obtidos pelas versões de VOGA e VOEA, com os *scores* obtidos por VOGA+ e VOEA+. Os resultados mostram que o uso de testes estatísticos para definir a população inicial pode trazer ganho no resultado final. Quando os *scores Bayesianos* obtidos pelas algoritmos que utilizam estes testes estatísticos não são melhores que os obtidos pelos algoritmos comuns, são iguais ou no mínimo bem próximos. Mesmo assim, o ganho de desempenho compensa seu uso principalmente para o VOGA-K2+ e VOEA+, como mostrado nas figuras 5.13, 5.14 e 5.15, onde o número de gerações necessárias para que os algoritmos alcançassem a convergência são comparados.

A Tabela 5.8 compara a MTCC obtida pelas versões de VOGAC e VOGAC+. Não há

diferença muito significativa em relação à taxa de classificação obtida pelos algoritmos. A figura 5.16 mostra que o VOGAC+ não trouxe melhoria quanto ao número de gerações necessárias para a convergência.

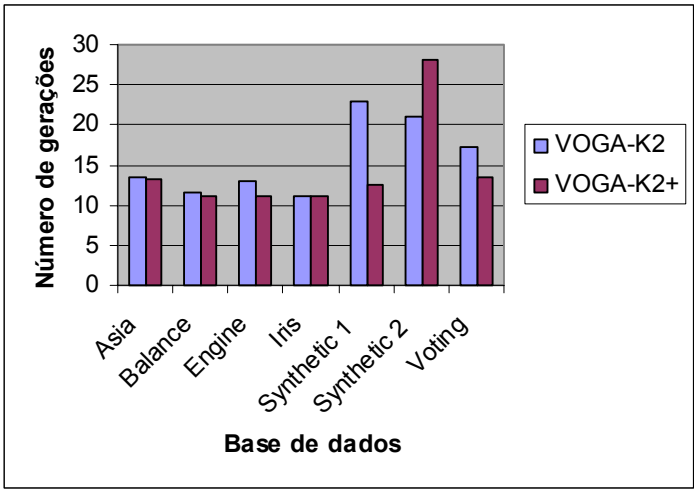


Figura 5.13. Número de gerações necessárias para a convergência dos algoritmos VOGA-K2 e VOGA-K2+.

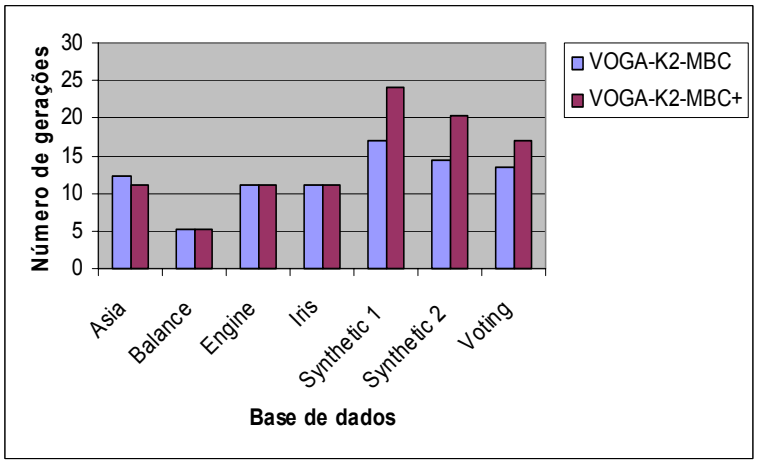


Figura 5.14. Número de gerações necessárias para a convergência dos algoritmos VOGA-K2-MBC e VOGA-K2-MBC+.

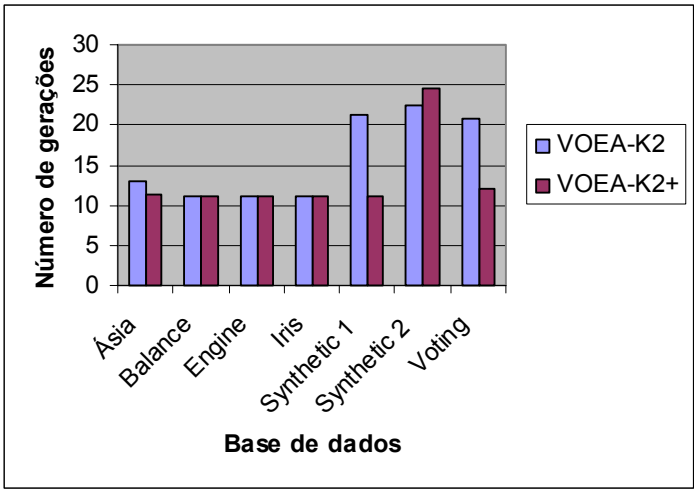


Figura 5.15. Número de gerações necessárias para a convergência dos algoritmos VOEA e VOEA+.

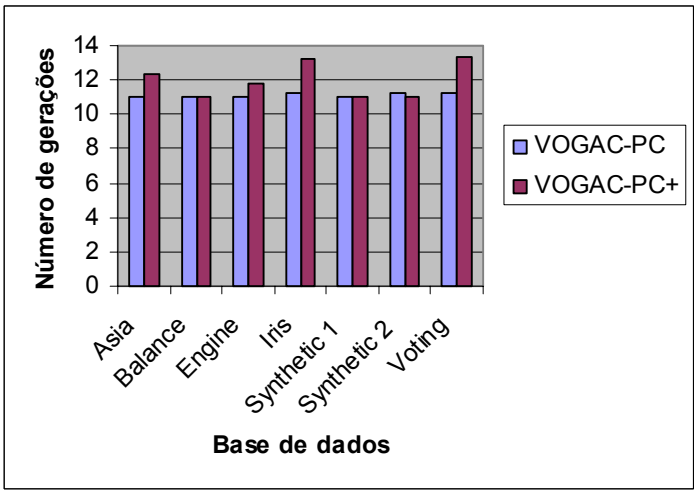


Figura 5.16. Número de gerações necessárias para a convergência dos algoritmos VOGAC-PC e VOGAC-PC+.

5.7.2 Alteração da Taxa de Cruzamento, Taxa de Mutação e Número de Gerações

Nos experimentos seguintes, foram usadas as bases de dados da Seção 5.3, porém com alterações nos seguintes parâmetros genéticos: taxa de cruzamento, taxa de mutação e número máximo de gerações sem melhorias. Os experimentos foram executados com os algoritmos de VOGA, VOGAC e VOEA, e utilizaram os valores definidos abaixo:

- A taxa de cruzamento foi alterada de 0.8 para 0.95;
- a taxa de mutação foi alterada de 30% dos genes dos cromossomos para 50%,

lembrando que este parâmetro não se aplica a VOEA;

- o número máximo de gerações sem melhorias foi aumentado de 10 para 20.

Nestes experimentos, os algoritmos VOGA, VOGAC e VOEA foram chamados de VOGA(1), VOGAC(1) e VOEA(1), respectivamente, para comparação com os resultados obtidos nos experimentos anteriores.

Adicionalmente, um novo conjunto de parâmetros foi testado, como descrito abaixo:

- A taxa de cruzamento foi alterada para 0.6;
- a taxa de mutação foi alterada para 10% dos genes dos cromossomos;
- o número máximo de gerações sem melhorias foi diminuído para 5.

Para estas simulações, os algoritmos de VOGA, VOGAC e VOEA foram chamados de VOGA(2), VOGAC(2) e VOEA(2), respectivamente. Os *scores Bayesianos* obtidos por VOGA, aplicando os novos valores definidos para os parâmetros genéticos, se encontram nas Tabelas 5.9 e 5.10 e são comparados com os *scores* obtidos nos experimentos da Seção 5.4.2. Na Tabela 5.11, são apresentadas as novas MTCC obtidas por VOGAC, comparadas com as anteriores, obtidas na Seção 5.5.1. E a Tabela 5.12, mostra os *scores* obtidos por VOEA.

Tabela 5.9. Média dos *scores Bayesianos* (função g) obtidos por VOGA-K2, VOGA-K2(1) e VOGA-K2(2).

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
VOGA-K2	-39741.37	-2254.79	-30386.79	-1835.88	**-76351.22**	**-79061.72**	-1548.83
VOGA-K2(1)	**-39740.98**	-2254.79	-30386.79	-1835.88	-76357.29	-79062.72	**-1547.339**
VOGA-K2(2)	-39741.6	-2254.79	-30386.79	-1835.88	-76356.18	-79065.30	-1551.345

Tabela 5.10. Média dos *scores Bayesianos* (função g) obtidos por VOGA-K2-MBC, VOGA-K2-MBC(1) e VOGA-K2-MBC(2).

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
VOGA-K2-MBC	-46502.08	-2254.79	-30456.43	-1835.88	**-86363.7**	-88961.21	-1809.37
VOGA-K2-MBC(1)	-46502.08	-2254.79	-30456.43	-1835.88	-86364.51	-88961.21	-1809.37
VOGA-K2-MBC(2)	-46502.08	-2254.79	-30456.43	-1835.88	-86364.63	-88961.21	-1809.37

Tabela 5.11. Média da Taxa de Classificação Correta (MTCC) obtida pelos algoritmos de VOGAC, VOGAC(1) e VOGAC(2).

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
VOGAC-PC	51.84	**86.30**	99.60	95.33	89.15	**93.43**	96.71
VOGAC-PC(1)	51.84	86.05	99.60	95.33	89.14	93.41	96.71
VOGAC-PC(2)	51.84	86.05	99.60	95.33	**89.16**	**93.43**	96.71

Tabela 5.12. Média dos *scores Bayesianos* (função g) obtidos por VOEA, VOEA(1) e VOEA(2).

	Asia	Balance	Engine	Iris	Synthetic 1	Synthetic 2	Voting
VOEA-K2	-39742	-2254	-30386	-1835	**-76352**	**-79060**	**-1549**
VOEA-K2(1)	**-39741**	-2254	-30386	-1835	-76355	-79061	**-1549**
VOEA-K2(2)	-39742	-2254	-30386	-1835	-76368	-79061	-1551

Os resultados apresentados nas Tabelas 5.9, 5.10, 5.11 e 5.12 revelam que os valores dos parâmetros genéticos modificados pouco influenciaram os algoritmos na busca de classificadores melhores. Percebe-se, pela Tabela 5.9, que aumentando os valores dos parâmetros genéticos, VOGA-K2 melhorou os *scores Bayesianos* apenas em dois domínios (*Asia* e *Voting*). Nos domínios *Synthetic 1* e *Synthetic 2*, os valores dos *scores Bayesianos* pioraram. A Tabela 5.10 mostra que VOGA-K2-MBC não obteve melhorias nos *scores Bayesianos* dos classificadores gerados, mantendo (com exceção de *Synthetic 1*) os mesmos valores encontrados anteriormente. Analisando a Tabela 5.11, nota-se que os algoritmos de VOGAC também não apresentaram melhorias significativas na MTCC, havendo poucas tendências de resultados melhores, tanto com os valores dos parâmetros mais altos, quanto mais baixos. A Tabela 5.12 mostra que VOEA-K2 melhorou apenas para o domínio *Asia*, ao utilizar os valores mais altos para os parâmetros.Assim, de maneira geral, a alteração dos parâmetros genéticos não trouxe mudanças significativas nas TCCs médias.

Um fato interessante, revelado nestes experimentos, refere-se à convergência dos algoritmos. Para as bases de dados com poucos atributos, os algoritmos não sofreram influência das taxas de cruzamento e mutação, pois obtiveram o melhor classificador já na primeira geração. Já as bases de dados com mais atributos, como *Synthetic 1*, *Synthetic 2* e *Voting*, comportaram-se de forma diferente, variando a convergência conforme as taxas aplicadas. Algumas foram mais

rápidas com os valores maiores, outras mais rápidas com os valores menores.

As bases de dados maiores também sofreram influência do número de gerações. Algumas tenderam a produzir resultados melhores com um valor maior de gerações sem melhorias. No entanto, não foi possível executar os algoritmos com um valor muito alto para este parâmetro, devido ao pouco tempo para os experimentos. Os experimentos futuros deverão testar, além de outros valores para os parâmetros genéticos, outras bases de dados maiores e com maior número de atributos.

5.8 Considerações Finais

Os resultados apresentados neste capítulo mostram que os métodos híbridos VOGA, VOGAC e VOEA são capazes de encontrar uma boa ordenação das variáveis e otimizar o aprendizado dos classificadores *Bayesianos*.

Ao utilizar bases de dados conhecidas e com poucos atributos (*Asia, Balance, Iris* e *Engine*), os métodos desenvolvidos induziram classificadores tão bons quanto os algoritmos clássicos de aprendizado de redes *Bayesianas* (K2 e PC). Já para bases de dados com um número maior de atributos (*Synthetic 1, Synthetic 2* e *Voting)*, os métodos conseguiram resultados melhores. Isto é interessante, visto que a dificuldade de encontrar uma ordenação de variáveis adequada varia conforme o número de variáveis da base de dados. Ou seja, em uma base de dados com poucos atributos existem relativamente poucas ordenações possíveis e assim, pode-se testar todas as possíveis ao invés de se utilizar um algoritmo genético. Já em bases de dados com um número elevado de atributos, o número de possíveis ordenação é também elevado (*n!*, onde *n* é o número de atributos da base) e, nestes casos, o uso de um algoritmo genético se torna mais adequado.

A geração de uma população inicial através de testes estatísticos se mostrou como uma boa heurística, melhorando a convergência dos algoritmos e produzindo classificadores melhores

em alguns experimentos. Entretanto, a alteração dos valores iniciais das taxas de cruzamento e mutação não influenciou significativamente os resultados, sendo necessários mais testes em bases de dados maiores e com mais atributos.

O capítulo seguinte apresenta alternativas de trabalhos futuros envolvendo os métodos híbridos desenvolvidos e relata algumas contribuições obtidas.

6 CONCLUSÕES

Este trabalho apresentou métodos híbridos desenvolvidos para buscar uma ordenação adequada de variáveis no processo de otimização do aprendizado de classificadores *Bayesianos* a partir de dados. Para isto, foram utilizadas técnicas de computação evolutiva, explorando as características de algoritmos evolucionários.

Classificadores *Bayesianos* podem ser induzidos com o uso de algoritmos de aprendizado de redes *Bayesianas*. No entanto, o espaço de busca para uma rede com *n* variáveis tem dimensão exponencial e são necessárias otimizações nos algoritmos para minimizar a complexidade computacional. A ordenação das variáveis é comumente adotada para a redução do espaço de busca do processo de aprendizado. Não há métodos automáticos computacionalmente eficientes, entretanto, na busca de uma ordenação adequada de variáveis para qualquer domínio de aplicação.

Neste trabalho, foram desenvolvidos métodos híbridos que aplicam computação evolutiva na busca de uma ordenação adequada de variáveis para o aprendizado de classificadores *Bayesianos*. Os métodos híbridos utilizam o conhecimento da variável classe para otimizar a cooperação entre algoritmos de aprendizado de redes *Bayesianas* e algoritmos evolucionários e foram denominados de VOGA, VOGAC e VOEA.

A abordagem VOGA utiliza um algoritmo genético para codificar as possíveis ordenações de variáveis do domínio em cromossomos e retornar a melhor após sua finalização. A função de aptidão, utilizada por VOGA para avaliar os cromossomos, foi definida em função do *score Bayesiano* calculado pelos algoritmos de aprendizado de redes *Bayesianas,* baseados em busca heurística. Os algoritmos de aprendizado, utilizados pelo VOGA nos experimentos deste trabalho, foram K2 e K2-MBC, gerando as versões VOGA-K2 e VOGA-K2-MBC.

O algoritmo K2-MBC foi desenvolvido neste trabalho para aprender classificadores *Bayesianos,* com o objetivo de melhorar o esforço computacional de VOGA. O K2-MBC foi baseado no algoritmo K2 e apresentou bons resultados de taxa de classificação, quando comparado a outros algoritmos, como K2 e *Naive Bayes*. Baseado nestes resultados, o K2-MBC foi utilizado pelo VOGA, o qual foi denominado de VOGA-K2-MBC.

A abordagem VOGAC diferencia-se da VOGA na definição da função de aptidão. Ao invés de usar o *score Bayesiano* 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. Nos experimentos descritos neste trabalho, VOGAC utilizou o algoritmo PC – baseado em independência condicional, gerando a versão: VOGAC-PC.

VOEA diferencia-se das duas outras abordagens por utilizar um algoritmo evolucionário, ao invés de um algoritmo genético, para realizar a busca da melhor ordenação. Em VOEA não é utilizado o operador de mutação e, desta forma, menos operações são necessárias na busca. Na versão implementada, foi utilizado o algoritmo K2 como algoritmo de aprendizado.

Os métodos híbridos foram avaliados em sete domínios de conhecimento. Os resultados obtidos mostraram-se satisfatórios, quando comparados com os algoritmos de aprendizado de redes *Bayesianas* clássicos, recebendo, como entrada, a ordenação original das variáveis nas bases de dados. Isto significa que os métodos conseguem buscar uma boa ordenação e otimizar o aprendizado dos classificadores *Bayesianos*, como fora proposto. Além disso, nos domínios com um maior número de variáveis, os métodos propostos se mostraram mais adequados do que nos domínios com um número menor de variáveis.

6.1 Contribuições Geradas

As maiores contribuições deste trabalho são os métodos e algoritmos híbridos propostos, implementados, analisados e avaliados. Desta forma pode-se destacar os métodos VOGA, VOGAC e VOEA com seus respectivos algoritmos VOGA-K2, VOGA-K2-MBC, VOGAC-PC e VOEA-K2, além do algoritmo K2-MBC. A proposta, implementação e desenvolvimento destes métodos e algoritmos envolveu o estudo e a apropriação de conhecimento nas áreas de aprendizado supervisionado de máquina (classificação); aprendizado de redes Bayesianas e classificadores Bayesianos com base em busca heurística e independência condicional; propagação de evidências em redes Bayesianas e classificadores Bayesianos; e computação evolutiva (com mais ênfase em algoritmos genéticos). Além disso, trouxe à luz algumas questões interessantes que serão melhor investigadas em trabalhos futuros como pode ser visto na Seção 6.2.

- **Artigos Publicados**

SANTOS, E. B.; HRUSCHKA JR., E.R.; 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, 2007. v. 1. p. 1.

SANTOS, E. B.; HRUSCHKA JR., E.R. 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.

HRUSCHKA JR., E.R., SANTOS, E. B., GALVÃO, S. D. C. O. Variable Ordering in the Conditional Independence Bayesian Classifier Induction Process: An Evolutionary Approach. In: 7th International Conference on Hybrid Intelligent Systems, 2007, Kaiserslautern, Los Alamitos: IEEE Press, 2007.

- **Artigo Submetido**

HRUSCHKA JR. E.R.; HRUSCHKA, E.R.; SANTOS, E.B.; EBECKEN, N.F.F., Towards Balancing Computational Efficiency and Accuracy in Bayesian Classifiers. International Conference on Tools with Artificial Intelligence (ICTAI) 2007.

6.2 Trabalhos Futuros

Pretende-se dar seqüência a este trabalho aplicando-se os métodos híbridos, já desenvolvidos, em domínios com bases de dados maiores e com um número maior de atributos. Além disso, pretende-se investigar a possibilidade de criação de operadores de cruzamento e mutação específicos para se obter maior eficiência nos métodos propostos, já que a variação dos operadores tradicionais não influenciou de maneira tão significativa o desempenho, conforme os resultados obtidos. Um outro trabalho futuro interessante é a inserção de técnicas auto-adaptativas nos algoritmos até então definidos, assim, os parâmetros genéticos poderão ser modificados durante a execução, de acordo com os indivíduos que forem surgindo, evitando que a qualidade da população caia, isto pode trazer uma melhoria nos resultados.

7 REFERÊNCIAS

[1] 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.

[2] BACK, T. **Evolutionary algorithms in theory and practice.** Oxford University Press, 1996.

[3] 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.

[4] 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.

[5] CAMPOS, L. M.; FERNADEZ-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**, 2002. p. 182-192.

[6] 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, 2000.

[7] CANO, R.; SORDO, C.; GUTIÉRREZ, J.M. Applications of bayesian networks in meteorology. In: GÁMEZ, J.A.; MORAL, S.; SALMERÓN, *A*. (Eds.). **Advances in Bayesian networks**. Springer Verlag, 2004. p. 309-327.

[8] CASTILLO, E.; GUTIERREZ, J.; HADI, A. **Expert systems and probabilistic network models.** New York: Springer-Verlag, 1996.

[9] CERQUIDES, J.; MANTARAS, R.L. Maximum a Posteriori Tree Augmented Naive Bayes Classifiers. **Discovery Science 2004**, p. 73-88, 2004.

[10] CESTNIK, B.; KONONENKO, I.; BRATKO I. A knowledge elicitation tool for sophisticated users. In: BRATKO, I.; LAVRAC, N. (Eds). **Progress in machine learning**. Wilmslow, U.K.: Sigma Press, p. 31-45, 1987.

[11] CHICKERING, D. M. Optimal structure identification with greedy search. **Journal of Machine Learning Research**, v. 3, p. 507-554, 2002.

[12] CHICKERING, D.; GEIGER, D.; HECKERMAN, D.E. **Learning Bayesian Networks is NP-Hard.** Microsoft: 1994. Research Technical Report MSR-TR-94-17.

[13] 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, 1996. p. 121-130.

[14] 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.

[15] COOPER G.; HERSKOVITZ, E. A Bayesian method for the induction of probabilistic networks from data. **Machine Learning,** v. 9, p. 309-347, 1992.

[16] 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.

[17] COZMAN, F. G. Generalizing variable elimination in Bayesian networks. In: WORKSHOP ON ARTIFICIAL INTELLIGENCE, AND COMPUTER VISION, 1 / WORKSHOP ON PROBABILISTIC REASONING IN ARTIFICIAL INTELLIGENCE, 1 / MEETING ON MULTI-AGENT COLLABORATIVE AND*.,* Atibaia, 2000. **Proceedings of the IBERAMIA / SBIA 2000.** São Paulo: Tec Art Editora, 2000. p. 27-32.

[18] 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.

[19] DUDA, R.; HART, P. **Pattern Classification and Scene Analysis**. New York: Wiley, 1973

[20] FRIEDMAN, N. et al. Bayesian network classifiers. **Machine Learning**, v. 29, p. 131-163, 1997.

[21] GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning.** Massachusetts: Addison-Wesley, 1989.

[22] GOOD, I. J. **The estimation of probabilities: an essay on modern Bayesian**

methods. M.I.T. Press, 1965.

[23] 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.

[24] HECKERMAN, D. **A Bayesian approach to learning causal.** Local: Microsoft Research, March, 1995, Notas: Technical Report MSR-TR-95-04.

[25] HECKERMAN, D. **A tutorial on learning Bayesian networks.** Local: Microsoft Research, Advanced Technology Division, 1995, Notas: Technical Report MSR-TR-95-06.

[26] 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.

[27] HOLLAND, J. H. **Adaptation in natural and artificial systems.** MIT Press, 1975.

[28] HRUSCHKA Jr., E. R.; EBECKEN, N.F.F. Towards efficient variables ordering for Bayesian Network Classifiers. Data Knowl. 2007.

[29] 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.

[30] 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.

[31] 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.

[32] KONONENKO; I. Semi-naïve Bayesian classifiers. In: WORKING SESSION ON LEARNING, PORTO. **6th Europ**. Portugal, p. 206-219, 1991.

[33] LACERDA, E. G. M; DE CARVALHO, A. C. P. L. F. Introdução aos algoritmos genéticos. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 19, 1999, **Anais**… Rio de Janeiro: EntreLugar. p. 51-126.

[34] LAM, W.; BACCHUS, F. Learning Bayesian belief networks: an approach based on the MDL principle. **Computational Intelligence**, v. 10, p. 269-293, 1994.

[35] 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.

[36] LARRAÑAGA, P. et al. Structure learning of Bayesian networks by genetic algorithms: a performance analysis of control parameters. **IEEE Journal on Pattern Analysis and Machine Intelligence**, v. 18, n. 9, p. 912-926, 1996.

[37] LARRAÑAGA, P; LOZANO, J. A. Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation, Kluwer Academic, 2001.

[38] LEE1, K, Y. et al. A-HEP: Adaptive Hybrid Evolutionary Programming for Learning Bayesian Networks. **Proceedings of International Conference GECCO'04***,* Late Breaking Papers, 2004.

[39] LEUNG, K. S.; LIANG, Y. Adaptive elitist-population based genetic algorithm. **Genetic and Evolutionary Computation Conference**, v. LNCS 2723, p. 1160–1171, 2003.

[40] LI, A. et al. An efficient structure learning method in gene prediction. **Proceedings of the International Conference on Neural Networks and Signal Processing**, p. 567-570, 2003.

[41] MICHALEWICZ, Z.; HINTERDING, R.; MICHALEWICZ, M. Evolutionary Algorithms. In: PEDRYCZ, W. (Ed**.**). **Fuzzy Evolutionary Computation.** Kluwer Academic, 1997. Cap 2.

[42] MITCHELL, M.; TAYLOR, C. E. Evolutionary Computation: an overview. In: **Annual Review of Ecology and Systematics**, v. 20, p. 593-616, 1999.

[43] MITCHELL, T. **Machine Learning**. McGraw Hill, 1997.

[44] MORALES, M.M.et al. A method based on genetic algorithms and fuzzy logic to induce Bayesian networks. **Proceedings of the Fifth Mexican International Conference in Computer Science**, p. 176-180, 2004.

[45] NEAPOLITAN, R. E. **Learning Bayesian networks.** New Jersey: Prentice Hall.

[46] NEWMAN, D.J. et al. UCI repository of machine learning databases. Irvine, CA:

University of California; Department of Information and Computer Science, 1998. Disponível em <http://www.ics.uci.edu/~mlearn/MLRepository.html>. Acesso em: 06/2006.

[47] PEARL, J. **Causality: models, reasoning, and inference**. Cambridge: University Press, Cambridge, UK, 2000.

[48] PEARL, J. Fusion propagation and structuring in belief networks. **Artificial Intelligence**, v. 29, n. 3, p. 241-288, 1986.

[49] PEARL, J. **Probabilistic reasoning in intelligent systems: networks of plausible inference**. San Mateo: Morgan Kaufmann, 1988.

[50] REBANE, G.; PEARL, J. The recovery of causal poly-trees from statistical data. In: KANAL, L.N.; LEVITT, T.S.; LEMMER, J.F. (Eds.). **Uncertainty in artificial intelligence 3**. Local: NorthHolland, Amsterdam, 1989.

[51] RESENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações.** Editora Manole.

[52] 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.

[53] 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: 1990. p. 295-308.

[54] SAHAMI, M. Learning limited dependence Bayesian classifiers. In: 2ND INT. CONF. KNOWLEDGE DISCOVERY IN DATABASES. **Proc**… Menlo Park, CA: AAAI Press, 1996. p. 334-338.

[55] 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.

[56] 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, 2007. v. 1.

[57] SHETTY, S.; SONG, M. Structure learning of Bayesian networks using a semantic genetic algorithm-based approach. **The 3rd International Conference Information Technology: Research and Education**, p. 454-458, 2005.

[58] SILVA, A. **Redes programadas geneticamente:** uma abordagem evolucionária à síntese de agentes autômatos. Tese de Mestrado, Universidade de Coimbra, 2000.

[59] SPEARS, W. et al. An overview of evolutionary computation. In: Proceedings of the European Conference on Machine Learning (ECML-93), LNAI, v. 667, p. 442-459, Springer-Verlag, 1993.

[60] SPIRTES, P.; GLYMOUR, C. An algorithm for fast recovery of sparse causal graphs. **Social Science Computer Review**, v. 9, p. 62-72, 1991.

[61] SPIRTES, P.; GLYMOUR, C.; SCHEINES, R. **Causation, prediction, and search.** New York: Springer-Verlag, 1993.

[62] 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..

[63] SYSWERDA, G. Schedule optimization using genetic algorithms. In: DAVIS, L. (Ed). **Handbook of genetic algorithms.** New York: Van Nostrand Reinhold, 1991. p. 332-349.

[64] TUCKER, A.; LIU, X.; OGDEN-SWIFT, A. Evolutionary learning of dynamic probabilistic models with large time lags. **International Journal of Intelligent Systems**, v. 16, n. 5, p. 621-645, John Wiley & Sons, 2001.

[65] VERMA, T.; PEARL, J. Equivalance 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.

[66] WANG, S; LI, S. Learning Bayesian networks by lamarckian genetic algorithm and its application to yeast cell-cycle gene network reconstruction from time-series microarray data. In: INTERNATIONAI WORKSHOP ON BIOLOGICALLY INSPIRED APPROACHES TO ADVANCED INFORMATION TECHNOLOGY, n. 1. **Lecture Notes in Computer Science**, v. 3141, p. 49-62, Lausanne, 2004.

[67] WITTEN, I. H.; FRANK, E. **Data Mining: Practical Machine Learning Tools and**

Techniques (Second Edition). Morgan Kaufmann, 2005.

[68] WONG, M. L.; LEE, S. Y.; LEUNG, K. S. A hybrid approach to discover Bayesian networks from databases using evolutionary programming. **Proceedings of the 2002 IEEE International Conference on Data Mining,** p. 498–505, 2002.

[69] WONG, M. L.; LEE, S. Y.; LEUNG, K. S. Data mining of Bayesian networks using cooperative coevolution. **Decision Support Systems**, v. 38, n. 3, p. 451-472, 2004.

[70] WONG, M. L.; LEUNG, K. S. An efficient data mining method for learning Bayesian networks using an evolutionary algorithm-based hybrid approach. **IEEE Transactions on Evolutionary Computation**, v. 8, n. 4, p. 378-404, 2004.

APÊNDICE A – Um Exemplo de Aplicação de Algoritmos Genéticos

Seja a função $f(x) = x^3$ definida para maximizar valores no intervalo de números inteiros entre 0 e 31. Uma possível solução deste problema de maximização através de algoritmo genético, consiste dos seguintes passos:

Passo 1: *Codificação dos valores que serão representados pelos cromossomos*: Neste exemplo, os valores presentes no intervalo de 0 a 31 são valores inteiros codificados em binário, isto é, 0 passa a ser 00000, 1 passa a ser 00001, e assim sucessivamente até o valor 31, que passa a ser 11111.

Passo 2: *Definição da população inicial*: Para este exemplo, é formada uma população de quatro cromossomos, escolhidos aleatoriamente: $x_1 = 01101 = 13$, $x_2 = 11000 = 24$, $x_3 = 01000 = 8$ e $x_4 = 10011 = 19$. A quantidade de cromossomos da população inicial poderia ser outra.

Passo 3: *Aplicação da função de aptidão para avaliar os cromossomos da população:* Aplicando a função $f(x) = x^3$, definida acima, tem-se: $f(x_1) = 2197$, $f(x_2) = 13824$, $f(x_3) = 512$, $f(x_4) = 6859$. Note que x_2 apresenta a melhor solução nesta geração, pois possui o melhor resultado entre os quatro cromossomos. Calculando a porcentagem de cada valor na soma total (23392), tem-se: $x_1 = 9\%$, $x_2 = 60\%$, $x_3 = 2\%$ e $x_4 = 29\%$.

Passo 4: *Seleção dos melhores cromossomos para formar a próxima geração:* Para a seleção dos cromossomos da nova geração, é escolhido aleatoriamente para este exemplo o método da roleta. A roleta será representada por uma reta (Figura 3.10), com ínicio em 0 e término em

100, dividida em quatro regiões que representam cada um dos cromossomos, de acordo com as aptidões encontradas no Passo 3. Assim, foram sorteados quatro números aleatórios que representam os cromossomos escolhidos para a próxima geração: 5, 22, 48, 89.

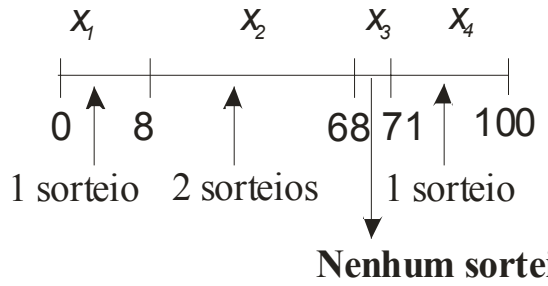


Figura 3.10. Reta numerada de 0 a 100%, dividida em quatro regiões, de acordo com as aptidões dos cromossomos.

Para os valores sorteados neste exemplo, x_1 é selecionado uma vez, pois apenas um dos números estava entre 0 e 8%; o cromossomo x_2 é selecionado duas vezes, pois teve dois números sorteados entre 9% e 68%; o cromossomo x_3 não será selecionado, pois nenhum dos números sorteados caiu entre 69% e 71%; e o cromossomo x_4 será selecionado uma vez, já que teve um número sorteado entre 72% e 100%. A nova geração após a seleção é: x_1 = *01101 = 13*, $\boldsymbol{x_2 = 11000 = 24}$, $\boldsymbol{x_3 = 11000 = 24}$ *e* x_4 *= 10011 = 19.* Como o x_3 da geração anterior tinha um valor de aptidão muito baixo, ele não foi selecionado e não possui nenhum representante na nova geração.

Passo 5: *Aplicação do operador de cruzamento:* é aplicado neste exemplo o operador de cruzamento de um ponto. Os cromossomos selecionados são colocados aos pares de maneira aleatória e um ponto de quebra para o cruzamento é escolhido aleatoriamente. Os cromossomos pareados são x_1 / x_2 e x_3 / x_4 e os pontos de quebra serão o dígito 4 e o dígito 2, respectivamente, conforme ilustrado na Figura 3.11.

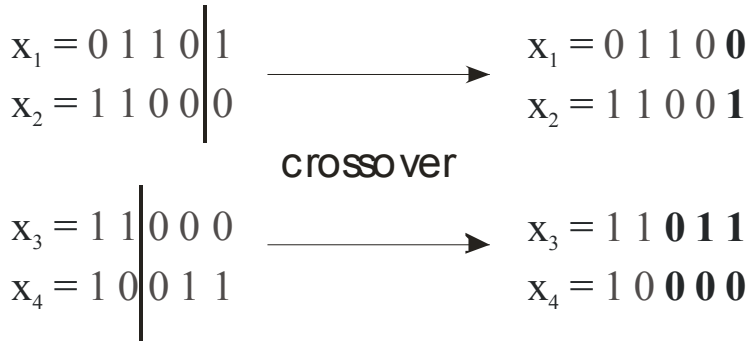


Figura 3.11. Cruzamento de um ponto realizado com os cromossomos do exemplo numérico.

Passo 6: *Aplicação do operador de mutação:* A mutação realizada neste exemplo envolve a troca de bits 0 por 1, e vice-versa. É sorteado um número entre zero e 20 (pois há 20 bits de informação, 4 cromossomos de 5 bits cada) para realizar a mutação. O bit sorteado foi o 13, ou seja, o terceiro bit do cromossomo x_3. Agora é sorteado um valor para determinar qual será o novo valor desse bit. O sorteio fornece o valor zero, isto é, o bit não mudará, pois seu valor original já é zero.

Depois de realizados os passos de cruzamento e mutação, a nova geração é avaliada novamente para determinar o valor de aptidão de cada cromossomo e o resultado é: $f(x_1) = 1728$, $f(x_2) = 15625$, $f(x_3) = 19683$, $f(x_4) = 4096$. Note que o cromossomo x_3 é o mais adaptado e seu valor é bem melhor do que o valor do melhor cromossomo (x_2) da geração anterior. O processo pode ser interrompido se o valor atingido for suficientemente bom, ou pode continuar por mais gerações até ser atingido o máximo da função $f(x)$ no intervalo escolhido.

Este exemplo ilustra de maneira didática a utilização de algoritmos genéticos em problemas de otimização numérica.