EVOLUÇÃO AUTOMÁTICA DE ALGORITMOS DE REDES BAYESIANAS DE CLASSIFICAÇÃO

ALEX GUIMARÃES CARDOSO DE SÁ

EVOLUÇÃO AUTOMÁTICA DE ALGORITMOS DE REDES BAYESIANAS DE CLASSIFICAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais – Departamento de Ciência da Computação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Gisele Lobo Pappa

Belo Horizonte

26 de fevereiro de 2014

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Sá, Alex Guimarães Cardoso de
S111e Evolução automática de algoritmos de redes bayesianas de classificação / Alex Guimarães Cardoso de Sá. — Belo Horizonte, 2014

xviii, 101 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais – Departamento de Ciência da Computação

Orientador: Gisele Lobo Pappa

1. Computação - Teses. 2. Classificação (Computadores) - Teses. 3. –Algoritmos de computador - Teses. I. Orientadora. II. Título.

51
9.6*61(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Evolução automática de algoritmos de redes bayesianas de classificação

ALEX GUIMARÃES CARDOSO DE SÁ

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROFA. GISELE LOBO PAPPA - Orientadora
Departamento de Ciência da Computação - UFMG

PROF. ADRIANO CÉSAR MACHADO PEREIRA
Departamento de Ciência da Computação - UFMG

PROF. MÁRCIO PORTO BASGALUPP
Departamento de Ciência e Tecnologia - UNIFESP

PROF. RENATO MARTINS ASSUNÇÃO
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 26 de fevereiro de 2014.

Agradecimentos

Primeiramente, gostaria de agradecer imensamente aos meus pais, José Roberto e Meire, e ao meu irmão, Lucas, por sempre estarem comigo e sempre me apoiarem em qualquer momento que eu precisasse.

À minha namorada, Larissa, pela compreensão e pelo carinho durante os momentos mais difíceis do mestrado, quando a conheci.

À toda minha família (tios, tias, primos, primas, avós, etc) pela presença constante em minha vida. Em especial, ao meu tio Ednaldo, pela ajuda durante o trabalho na parte da aplicação dos métodos estatísticos.

Aos amigos e amigas pelas boas conversas, pela amizade e pelo companheirismo. Muitos estiveram longe pela distância (Lavras, Ijaci, São João del Rei, Conselheiro Lafaiete, Rio de Janeiro, São Paulo, Uberlândia e Venda Nova do Imigrante), mas podem ter certeza que sempre estiveram e estão comigo.

Aos meus novos amigos e colegas de Belo Horizonte, que conheci logo que comecei o mestrado. Foram anos que tiveram momentos de estresse, sem vocês para sair ou mesmo só para conversar, tenho certeza que não valeria a pena. Especialmente, agradeço ao Luiz Otávio, com quem tive maior amizade e companheirismo profissional nesses últimos tempos.

À minha orientadora, Gisele L. Pappa, pela oportunidade, orientação e, principalmente, pela paciência em me ajudar em todo o trabalho.

Aos meus antigos orientadores (Tales Heimfarth, Braulio A. de Mello, Marluce R. Pereira e Thiago S. Rodrigues) pelo incentivo e motivação à pesquisa.

Aos professores da UFMG com que tive mais contato, sejam pelas aulas ou pelo contato em algum projeto.

Por fim, à CAPES, à UFMG e ao Laboratório e-Speed pelo apoio financeiro, técnico e científico.

Resumo

Quando nos deparamos com um novo problema de classificação, selecionar o classificador mais adequado para a tarefa é geralmente um desafio. Isso porque cada base de dados tem características diferentes, que podem tornar um classificador mais apropriado que outro. A área de meta-aprendizado surgiu para resolver este tipo de problema: selecionar um algoritmo para uma determinada base de dados de acordo com um conjunto de meta-dados.

Nessa direção, este trabalho propõe uma nova abordagem para construir um algoritmo adaptado à(s) base(s) de dados da aplicação de interesse. Mais especificadamente, propõe-se um algoritmo evolucionário (AE) para evoluir automaticamente algoritmos de Redes Bayesianas de Classificação (RBCs). RBCs são modelos de classificação interessantes por serem robustos à falta de dados e incerteza, além de gerarem modelos de classificação interpretáveis.

O AE proposto recebe como entrada uma lista de componentes principais de algoritmos de RBCs e uma (ou um conjunto de) base(s) de dados de entrada. Com esses dois elementos, o AE testa diferentes combinações dos componentes, gerando um algoritmo personalizado para aqueles dados.

Para validar o AE, os experimentos foram divididos em três partes principais: (**i**) testes do método proposto em execuções direcionadas a bases de dados específicas, (**ii**) testes em execuções direcionadas a conjuntos de bases dados semelhantes e (**iii**) testes com conjuntos bases de treino e teste distintos. Para a primeira parte, 15 bases de dados da UCI foram escolhidas para testes em bases específicas a fim de gerar algoritmos personalizados para as mesmas. Já para as outras duas partes, concentrou-se na aplicação do método proposto em conjuntos de bases de dados. Nesse caso, 20 bases de dados com particularidades distintas foram selecionadas com o intuito de realizar um agrupamento sobre as mesmas, podendo assim criar os diferentes cenários dos experimentos onde exista conjuntos de bases de treinamento e teste.

Testes sobre o AE foram realizados considerando as três partes dos experimentos e os resultados foram comparados separadamente com uma busca gulosa e, em seguida,

com três algoritmos estado-da-arte de RBCs (Naïve Bayes, TAN e K2). Os resultados mostraram que os algoritmos gerados são competitivos com aqueles dos métodos do estado-da-arte, e que na maioria dos casos o uso de algoritmo evolucionário em invés de uma simples busca gulosa melhora estatisticamente os resultados.

Palavras-chave: Redes Bayesianas de Classificação, Evolução Automática, Componentes, Problema de Classificação, Algoritmo Evolucionário.

Abstract

When faced with a new machine learning problem, selecting which classifier is the best to perform the task at hand is a very hard problem. The reason for this is the nature of the data used by the classifier, which can differ abruptly from one set to another, consequently affecting the classification outcome. In other words, the same classifier can not be adapted to different types of data. Most solutions proposed in the literature are based on meta-learning, and use meta-data about the problem to recommend an effective algorithm to solve the task.

This work proposes a new approach to this problem: to build an algorithm tailored to the application problem at hand. More specifically, we propose an evolutionary algorithm (EA) to automatically evolve Bayesian Network Classifiers (BNCs). The method receives as input a list of the main components of BNC algorithms, and uses an EA to encode these components. Given an input dataset (or a group of datasets), the method tests different combinations of components and returns the "best" BNC algorithm to that specific application domain.

For testing, we divided the experiments in three main parts: (**i**) tests in specific datasets domains; (**ii**) tests directed to sets of similar datasets; (**iii**) tests directed to sets of distinct datasets. For the first part, 15 UCI datasets were chosen to evaluate the proposed approach and generate tailored algorithms for these datasets. The other two parts focused on applying the EA on sets of datasets. In this case, 20 datasets with distinct characteristics were selected in order to cluster them and, thus, create different experiment scenarios.

Tests were performed on the AE considering the three parts of experiments and results were compared separately with a greedy search method and, then, with three state-of-art BNC algorithms (Naïve Bayes, TAN and K2). Results showed that the generated BNC algorithms are competitive with those of the state-of-art methods, and in most cases the use of an evolutionary algorithm, rather than a simple greedy search, improved statistically the results.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Redes Bayesianas de Classificação (RBCs) são consideradas ferramentas estatísticas robustas e precisas para solucionar problemas de classificação de dados. Elas possuem fundamentação teórica derivada de Rede Bayesianas e assumem em seu modelo que existem relações de causa e efeito entre os atributos previsores da bases de dados [Friedman et al., 1997]. Uma RBC é representada por um grafo acíclico direcionado e uma tabela de probabilidades condicionais para cada nó. Um nó representa um atributo da base de dados, as arestas dependências entre eles, e as tabelas são consideradas parâmetros da rede.

Existem três motivações principais para se trabalhar com RBCs [Heckerman, 1997]: (i) RBCs codificam as dependências entre todas as variáveis do problema e, com isso, estão prontas para lidar com uma eventual falta de dados; (ii) RBCs podem ser utilizadas para aprender relacionamentos causais e, portanto, podem ser usadas para ganhar entendimento sobre o domínio do problema e predizer as consequências de eventos; (iii) RBCs são modelos gráficos os quais podem ser interpretados por especialistas de domínio.

Mesmo RBCs sendo consideradas abordagens precisas por tratarem questões de incerteza e dependências entre atributos, existe uma infinidade de algoritmos de RBCs disponíveis na literatura, cada um com suas particularidades. Portanto, justifica-se questionar: dada uma (ou um conjunto de) base de dados, qual é o melhor algoritmo para gerar o melhor modelo de classificação? Em outras palavras, quando nos deparamos com um novo problema de aprendizado, qual classificador devemos escolher para executar essa nova tarefa (da melhor maneira possível)?

Para tentar responder essa questão, surgiu a área de meta-aprendizado, que pode ser definida de forma abrangente como o processo de aprender a aprender [Vilalta & Drissi, 2002]. De acordo com Giraud-Carrier et al. [2004], meta-aprendizado tam-

bém pode ser descrito como o processo de aquisição de conhecimento que relaciona o desempenho dos algoritmos com as características dos problemas. A área de meta-aprendizado é de extrema relevância visto que, de acordo com o teorema do *No-Free Lunch* [Wolpert & Macready, 1997], dois algoritmos quaisquer têm em média o mesmo desempenho se forem considerados todos os problemas possíveis.

A diferença entre técnicas de aprendizado tradicional e meta-aprendizado está no nível de especifidade dos algoritmos. Enquanto o aprendizado tradicional foca em aprender sobre uma base de dados específica, a área de meta-aprendizado é bem mais geral, almejando adquirir experiência sobre diferentes tipos de bases de dados.

Existem basicamente dois tipos de técnicas de meta-aprendizado: seleção de algoritmos e combinação de modelos. No primeiro caso, o algoritmo tenta, a partir de informações da base de dados, recomendar um algoritmo de aprendizado (neste caso, de classificação), baseando-se nas características das bases. Por exemplo, pode-se recomendar um algoritmo por uma regra: *se (número de classes da base de dados maior que dois) então use um KNN.*

Por outro lado, a combinação de modelos tenta solucionar o problema de aprender a respeito de uma base de dados a partir da combinação de algoritmos de aprendizado. Assim, utiliza-se geralmente diferentes algoritmos para aprender sobre diferentes partes dos dados. Esses diferentes algoritmos podem se referir tanto à parametrização de um mesmo algoritmo, quanto à distinção do próprio método. As técnicas de *Bagging* [Breiman, 1996], *Boosting* [Freund & Schapire, 1995] e *Stacking* [Wolpert, 1992] fazem parte desse tipo de aprendizado.

Uma terceira forma de meta-aprendizado proposta mais recentemente é a geração de algoritmos de aprendizado [Barros et al., 2012; Pappa & Freitas, 2009b; Yao, 1999]. Nesse caso, um algoritmo é utilizado para combinar as partes essenciais (ou componentes) de algoritmos de aprendizado específicos (*e.g.*, árvores de decisão, regras de indução, etc.) que gerem um modelo de representação particular. Como exemplos de possíveis componentes estão a heurística que o algoritmo utiliza para percorrer o espaço de soluções, como é feita a ponderação para direcionar a busca do algoritmo, condição de parada, entre outros.

Dois modos são amplamente utilizados para a geração de algoritmos: geração por seleção e geração por construção. Enquanto a geração por seleção escolhe componentes já existentes pertencentes a métodos potencialmente diferentes e realiza sua combinação, dando origem a um algoritmo, a geração por construção combina esses componentes com primitivas de algoritmos (laços, condicionais, etc) e componentes ainda não testados para construir novos métodos de aprendizado.

As primeiras abordagens baseadas na geração de algoritmos focaram na evolução

de redes neurais artificiais [Yao, 1999]. Após redes neurais, abordagens para construir algoritmos de indução de regras [Pappa & Freitas, 2009b,c] e árvores de decisão [Barros et al., 2012] também foram propostos. Aqui propomos uma nova abordagem, que gera redes Bayesianas de classificação.

Em todas as abordagens mencionadas anteriormente, algoritmos evolucionários foram utilizados como métodos de busca para construir algoritmos de aprendizado personalizados a bases de dados. Algoritmos evolucionários [Eiben & Smith, 2003] são métodos estocásticos baseados nas teorias de evolução e sobrevivência do indivíduo mais adaptado de Darwin. Eles são métodos de busca global, frequentemente utilizados para resolver tarefas na área de aprendizado de máquina e serão utilizados neste trabalho para gerar automaticamente algoritmos de redes Bayesianas de classificação, como detalhado na próxima seção.

1.1 Objetivos

O objetivo deste trabalho é propor um algoritmo evolucionário para gerar de forma automática o "melhor" algoritmo de RBC, de acordo com uma (ou um conjunto de) base(s) de dados previamente selecionada(s). O algoritmo evolucionário proposto é qualificado como um algoritmo de meta-aprendizado que realiza geração de algoritmos por seleção.

A Figura 1.1 ilustra o método proposto. Um algoritmo evolucionário recebe como entrada a(s) base(s) de dados e o conjunto de algoritmos de redes Bayesianas de classificação, organizados por diversos tipos de componentes. Tendo como entrada esses elementos, o algoritmo evolucionário desenvolverá um algoritmo (ou vários algoritmos) de RBC específico (s) para essa(s) base(s) de dados. A ideia principal é que esse algoritmo de RBC possa criar um modelo mais adequado para a tarefa de classificação sendo considerada. A adequação do modelo de classificação é medida de acordo com métricas de qualidade de classificadores, como precisão e revocação.

Na Figura 1.1 o modelo gerado possui apenas três atributos previsores (A1, A2 e A3) e o atributo classe (C). As setas no modelo gerado indicam as relações causais entre os atributos da(s) base(s) de dados. Esse é apenas um simples exemplo que deve ser extrapolado para contextos mais complexos, que ocorrem com maior frequência em problemas reais.

Quatro objetivos específicos foram também vislumbrados pelo presente trabalho:

1. Revisão bibliográfica sobre Redes Bayesianas e as diversas técnicas de aprendizado de Redes Bayesianas Gerais e Específicas à Classificação;

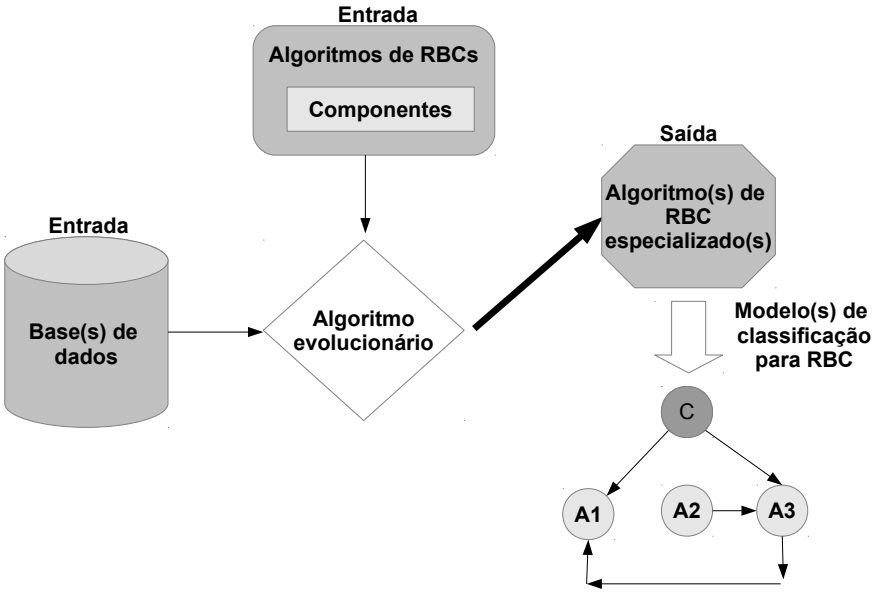


Figura 1.1: Evolução automática de algoritmos de RBCs.

2. Identificação e mapeamento dos componentes de algoritmos de RBCs. Componentes podem ser definidos como partes essenciais dos algoritmos de RBCs, como por exemplo: módulos que alteram o funcionamento do método de busca de RBCs, métricas que ponderam a qualidade a RBC criada e parâmetros que determinam a profundidade da busca a ser realizada;

3. Identificação dos métodos mais apropriados para comparação com o método proposto;

4. Avaliação do método proposto em um conjunto de bases de dados relevantes da literatura, considerando as diferenças de características entre as mesmas.

1.2 Principais Contribuições

De acordo com os objetivos listados na seção anterior, as principais contribuições deste trabalho são:

- Uma abordagem baseada em um algoritmo evolucionário para trabalhar com a evolução de algoritmos de RBCs. A ideia principal dessa abordagem é gerar um algoritmo de RBC personalizado às bases de dados de entrada.

- Identificação de um conjunto de componentes comuns aos algoritmos de aprendizado de Redes Bayesianas de Classificação.
- Desenvolvimento e adaptação de um conjunto de algoritmos e técnicas de RBCs em um *framework* de aprendizado de máquina.

1.3 Organização do Texto

O restante deste trabalho encontra-se dividido da seguinte forma: o Capítulo 2 apresenta conceitos básicos referentes a redes Bayesianas, algoritmos evolucionários e também discute os trabalhos relacionados. O Capítulo 3 descreve de forma geral o método de evolução de RBCs, foco do presente trabalho. O Capítulo 4 apresenta os resultados encontrados pelo método proposto com discussões sobre o significado de cada um deles. No Capítulo 5 são feitas as conclusões deste trabalho, apresentando também os próximos passos desta pesquisa.

Capítulo 2

Revisão Bibliográfica

Este capítulo tem por objetivo definir os conceitos fundamentais de Redes Bayesianas (RBs) e redes Bayesianas de classificação (RBCs). Ele também apresenta um resumo dos principais componentes presentes nos algoritmos de RBCs propostos na literatura, que será utilizado no Capítulo 3. Por último, ele introduz os conceitos básicos de Algoritmos Evolucionários e revisa trabalhos relacionados a evolução automática de algoritmos de classificação.

2.1 Redes Bayesianas

De acordo com Cheng & Greiner [1999], uma rede Bayesiana é uma ferramenta robusta para descrever/representar o conhecimento ou tirar conclusões a respeito de suas propriedades, levando sempre em conta condições de incerteza.

Uma RB é um grafo acíclico direcionado (DAG, do inglês *Directed Acyclic Graph*) com uma distribuição condicional de probabilidade para cada nó da rede. A estrutura do DAG contém nós representando o domínio das variáveis e arcos entre esses nós representando as suas dependências probabilísticas. Bari [2011] acrescenta que uma RB representa a distribuição de probabilidade conjunta entre um (grande) número de variáveis, e permite executar **inferência probabilística** com essas variáveis. Na construção de redes Bayesianas de bases de dados, usa-se nós para representar os atributos da base. Neste trabalho, o conceito de nós, variáveis e atributos irão ser utilizados como sinônimos.

Heckerman [1997] mostra as principais vantagens de se usar uma rede Bayesiana para modelagem dos dados. Em primeiro lugar, uma RB é um modelo que codifica as dependências entre todas as variáveis, estando pronta para lidar com situações em que alguns dos dados são faltantes ou desconhecidos. Segundo, uma RB pode ser utilizada

para aprender a respeito de relacionamentos causais e, portanto, ganhar entendimento sobre o domínio de um problema e predizer as consequências de uma intervenção (modificação do conhecimento sobre as variáveis). Por ser um modelo que possui semântica causal e probabilística, uma RB é a representação ideal para combinar conhecimento *a priori* (que vem de uma forma causal) e dados. Por fim, métodos estatísticos Bayesianos em conjunto com RBs oferecem uma abordagem eficaz para evitar o superajuste (*overfitting*) dos dados.

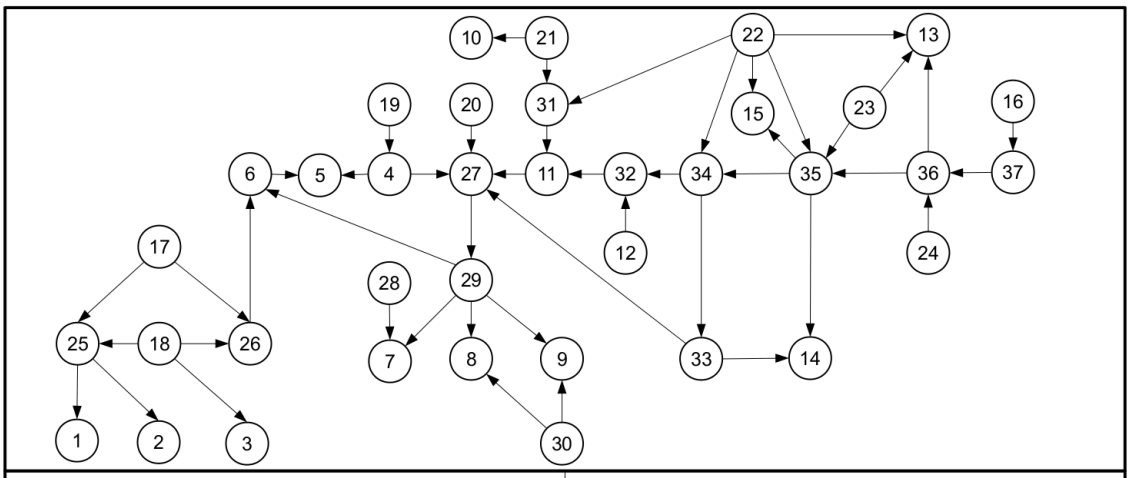
Como mencionado por Daly et al. [2011], é válido saber que redes Bayesianas (*Bayesian networks*) são frequentemente conhecidas por outros nomes, incluindo: modelos gráficos recursivos (*recursive graphical models*), redes Bayesianas de crença (*Bayesian belief networks*), redes de crença (*belief networks*), redes probabilísticas causais (*causal probabilistic networks*), redes causais (*causal networks*), diagramas de influência (*influence diagrams*), entre outros.

Geralmente, RBs são usadas com dois propósitos principais: (1) indução de redes por meio de exemplos; (2) tarefa de classificação de dados. Para o primeiro caso já existe um estudo bem consolidado, conforme será visto no texto adiante. A segunda finalidade para RBs ainda é um assunto que merece bastante estudo, dadas as vantagens supracitadas.

A Figura 2.1 [Beinlich et al., 1989] será utilizada para diferenciar esses dois propósitos a partir de um exemplo. A rede ALARM (*A Logical Alarm Reduction Mechanism*) é um sistema de diagnóstico médico com 37 nós e 46 arcos para monitorar pacientes. A descrição de cada nó está abaixo da figura. Existem três tipos de nós: diagnóstico, medidas informativas e intermediários (onde medidas não podem ser calculadas diretamente). Essa rede tornou-se uma forma bem comum de testar a qualidade de um algoritmo de indução de redes via dados.

Para indução de redes, o objetivo é dizer a probabilidade de ocorrer um evento dado que outros eventos ocorram. Na Figura 2.1, pode-se querer saber quais são as chances de ocorrer um diagnóstico de embolia pulmonar (nó 21) dadas as medidas do desvio (*shunt* − nó 31) e da pressão da artéria pulmonar (nó 10). Mas isso não impede que se realize outros tipos de diagnósticos na rede, tais como hipovolemia (nó 17).

Já no caso da tarefa de classificação existiria apenas um nó de diagnóstico na figura, que seria o nó classe, e esse nó afirmaria qual é o diagnóstico de um paciente dentre os possíveis, dados os diferentes tipos de medidas informativas desse paciente, como: pressão sanguínea e arterial, saída cardíaca, dentre outros. Assim, nesse segundo caso a variável de interesse é única, enquanto no primeiro ela pode se referir a qualquer nó da rede.



1- Pressão venosa central
2- Pressão capilar pulmonar
3- Histórico de insuficiência do ventrículo esquerdo
4- Resistência periférica total
5- Pressão sanguínea
6- Débito cardíaco
7- Frequência cardíaca obtida do monitor de pressão sanguínea
8- Frequência cardíaca obtida do eletrocardiograma
9- Frequência cardíaca obtida do oxímetro
10- Pressão da artéria pulmonar
11- Saturação de oxigênio no sangue arterial
12- Fração de oxigênio no gás inspirado
13- Pressão de ventilação
14- Conteúdo de dióxido de carbono do gás expirado
15- Volume (respiratório) por minuto, medido
16- Volume (respiratório) por minuto, calculado
17- Hipovolemia
18- Insuficiência do ventrículo esquerdo
19- Anafilaxia
20- Anestesia ou analgesia insuficiente
21- Embolia pulmonar
22- *Status* de intubação
23- Tubo de ventilação torcido
24- Tubo de ventilação desconectado
25- Volume diastólico final do ventrículo esquerdo
26- Volume sistólico
27- Nível de catecolamina
28- Erro na leitura da frequência cardíaca devido ao baixo débito cardíaco
29- Frequência cardíaca verdadeira
30- Erro na leitura da frequência cardíaca devido ao dispositivo de eletrocauterização
31- Desvio
32- Saturação de oxigênio da artéria pulmonar
33- Conteúdo de dióxido de carbono arterial
34- Ventilação alveolar
35- Ventilação pulmonar
36- Ventilação medida no tubo endotraqueal
37- Ventilação por minuto medida no respirador

Figura 2.1: A Rede ALARM – Figura adaptada de Beinlich et al. [1989].

2.1.1 Tipos de Conexões

RBs representam as relações entre diversas variáveis de um modelo, e portanto deve-se estudar os tipos de relações por meio dos tipos de conexões entre nós da RB.

As conexões revelam muito sobre a natureza das variáveis do modelo probabilístico, ou seja, a dependência entre duas ou mais variáveis. Assim, duas variáveis são dependentes (a observação de uma variável influencia na observação da outra) caso as arestas entre as mesmas estejam direcionadas. O fluxo de influência pode também não ser totalmente explícito, podendo acontecer de forma condicional. Por exemplo, duas variáveis podem ser dependentes dado o resultado de uma terceira variável (ou um conjunto de variáveis).

De forma oposta, pode-se analisar também quando duas variáveis são dita independentes, isto é, resultados de uma variável não influenciam no resultado de uma segunda. A independência condicional também ocorre com frequência em redes Bayesianas.

Existem basicamente três tipos de conexões em redes Bayesianas: serial, divergente e convergente. O trabalho de Krieg [2001] descreve apropriadamente cada tipo de conexão, ressaltando que o entendimento das conexões é importante porque, a partir delas, também se pode entender o conceito de d-separação, que é utilizado por alguns dos algoritmos que serão vistos neste trabalho.

O primeiro tipo de conexão é a serial, exemplificado pela Figura 2.2. Nessa figura é visto o fluxo de influência entre as variáveis A, B e C. Qualquer evidência sobre A influencia B que, por sua vez, afeta C. Se o estado de B é informado, então o canal de influência entre A e C é fechado, tornando-as variáveis d-separadas. Em outras palavras, A e C são separadas por B, ou condicionalmente independentes dado B.

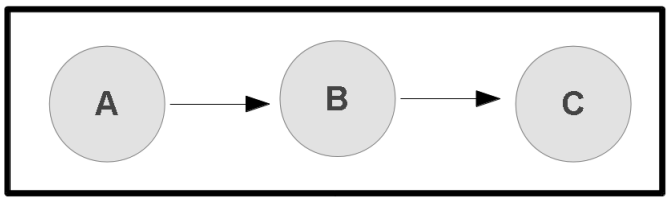


Figura 2.2: Conexão serial entre variáveis.

O tipo de conexão divergente é o segundo dentre as possíveis categorias de conexões de RBs. A Figura 2.3 apresenta um exemplo em que as variáveis B e C divergem de A. Pode ser dito que a influência será transmitida entre todos os filhos de A, a menos que o estado de A seja conhecido. Portanto, se a variável A é instanciada, B e C são d-separados, ou condicionalmente independentes dado A.

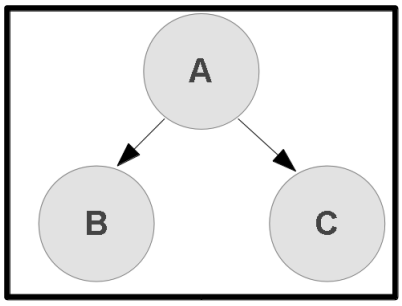


Figura 2.3: Conexão divergente entre variáveis.

Por fim, a conexão convergente é ilustrada na Figura 2.4. Nessa figura, as variáveis B e C convergem para A, ou seja, a variável A depende de B e C. Caso não se saiba nada sobre A, exceto o que pode ser concluído a partir de B e C, o que determina que independência marginal entre B e C (que são "pais" ou nós precedentes causais de A) seja assumida pelo modelo. Portanto, B e C não são d-separados quando se conhece o estado de A, isto é, B e C são condicionalmente dependentes dado A.

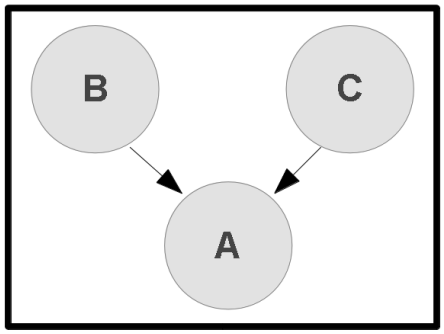


Figura 2.4: Conexão convergente entre variáveis.

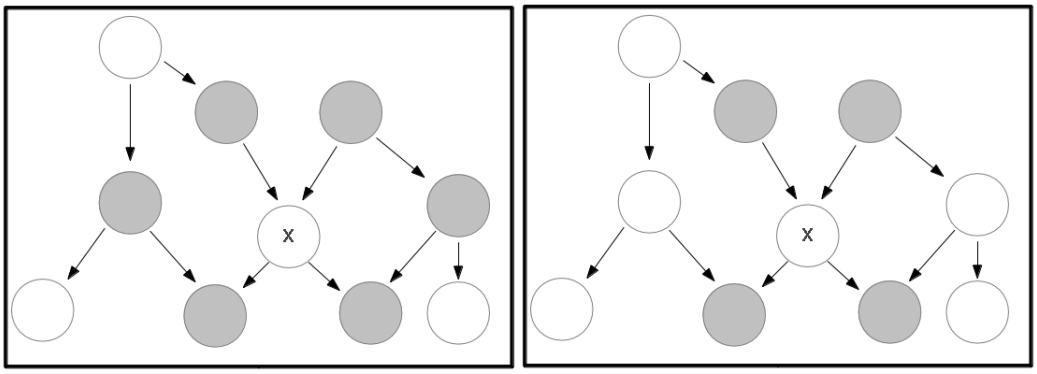
Os exemplos anteriores para tipos de conexões possuem apenas três variáveis de interesse, conduzindo a casos simples de nós d-separados. Entretanto, em domínios reais, o número de variáveis é bem superior, o que torna o processo de examinar se duas variáveis são d-separadas mais complexo. Como já mencionado, a aplicação da d-separação em redes Bayesianas é utilizada por alguns algoritmos presentes na literatura para o aprendizado da estrutura de uma RB.

Em resumo, caracteriza-se o conceito de d-separação por meio da seguinte regra: duas variáveis A e B em uma rede Bayesiana são d-separadas se, para todos os caminhos entre A e B, existe uma variável intermediária I tal que a conexão é serial ou divergente e o estado de I é conhecido; ou a conexão é convergente e não se tem conhecimento a respeito de I e seus descendentes.

A definição de d-separação não é a única utilizada pelos algoritmos descritos por este trabalho. Existem trabalhos que exploram outros conceitos como Cobertura de Markov ou Cobertura Aproximada de Markov, que serão colocados em contexto a seguir.

O conceito de Cobertura de Markov (CM) é descrito e utilizado por Santos et al. [2011] para supor a independência de um nó do grafo (no caso desse artigo, o nó de classificação) em relação aos demais nós da rede. Pode-se dizer que um nó é condicionalmente independente de todos os outros nós de um grafo dados seus pais, filhos (nós descendentes pelo fluxo causal) e pais de seus filhos (isto é, sua CM). A Figura 2.5a apresenta um exemplo. Nela os vértices sombreados são a Cobertura de Markov do vértice X.

Já a Cobertura Aproximada de Markov (CAM), também presente em Santos et al. [2011], é apenas uma variação da Cobertura de Markov, que deve ser aplicada separadamente da sua versão original. É dito que um nó é condicionalmente independente de todos os outros nós de um grafo dados seus pais e seus filhos. A Figura 2.5b ilustra a CAM pelo mesmo exemplo da Figura 2.5a. Nela os vértices sombreados são



(a) Cobertura de Markov para o vértice X.

(b) Cobertura Aproximada de Markov para o vértice X.

Figura 2.5: Cobertura de Markov e sua versão aproximada.

a Cobertura Aproximada de Markov do vértice X.

A utilização da CM e/ou CAM para os trabalhos que serão descritos a seguir é motivada pela diminuição do custo computacional dos algoritmos. Quando se aplica CM e/ou CAM reduz o escopo algoritmo, ou seja, diminui-se a quantidade de variáveis a serem exploradas por tal algoritmo.

2.1.1.1 Direções de Pesquisa

Independente do tipo de conexões da rede Bayesiana, trabalhos a respeito deste tema podem caminhar para diferentes direções.

Como relatado em Heckerman [1997] e Daly et al. [2011], trabalhos sobre redes Bayesianas geralmente abordam três vertentes: aprendizado da estrutura, aprendizado dos parâmetros e inferência probabilística. Essas vertentes podem ser combinadas ou utilizadas isoladamente. Mesmo que o presente trabalho tenha como objetivo o aprendizado da estrutura da rede e posterior aplicação em um classificador, é importante ver do que se trata os demais tipos de trabalhos, mesmo que de forma geral.

1. **Aprendizado da estrutura:** Aprender a estrutura de uma rede Bayesiana pode ser considerado um exemplo específico de um problema geral de selecionar um modelo probabilístico que melhor explica um determinado conjunto de dados. Nas próximas duas subseções, este tópico será discutido com uma maior ênfase por sua importância para o presente trabalho.

2. **Aprendizado dos parâmetros:** É também considerada uma tarefa importante dentro do contexto de aprendizagem de estrutura, pois muitos algoritmos de aprendizagem da estrutura estimam parâmetros após a criação da estrutura da

RB (modelo). O aprendizado de parâmetros geralmente consiste em aprender a partir dos dados as tabelas de probabilidades condicionais que estarão presentes em cada nó da RB.

3. **Inferência:** Inferência em RBs geralmente possui dois significados: (1) descobrir a probabilidade de uma variável estar em um certo estado, dado que outras variáveis estão definidas para um conjunto restrito de valores; (2) determinar os valores de um dado conjunto de variáveis que melhor explica a razão de um conjunto de outras variáveis estarem configuradas com certos valores. O processo de inferência é frequentemente utilizado dentro de sub-rotinas de problemas de aprendizado da estrutura da RB.

2.1.2 Aprendizado da Estrutura de Redes Bayesianas Gerais

Esta subseção demonstra como construir a estrutura de uma rede Bayesiana geral por meio de exemplos. Esse tópico se torna importante para o escopo deste trabalho, porque muitos dos algoritmos de redes Bayesianas gerais são utilizados como inspiração para a criação de classificadores Bayesianos específicos. Além disso, muitos desses métodos podem ser aplicados (sem grandes modificações) para criação de classificadores não-específicos.

Será utilizada a taxonomia proposta nos trabalhos de Cheng et al. [2002] e de Daly et al. [2011]. Ambas literaturas diferenciam os trabalhos em categorias como, por exemplo, o tipo de modelo criado (árvore, poli-árvore ou um grafo geral), se a ordenação causal dos nós é requirida como entrada para o método e o tipo do algoritmo para criar a estrutura (Busca e Pontuação, Restrições, Programação Dinâmica e Abordagens Híbridas). Cada categoria está brevemente descrita nos itens a seguir:

- **Tipo de modelo criado**: São três tipos principais: árvores, poli-árvores e grafos. Árvores e poli-árvores são estruturas Bayesianas que contêm um único caminho entre dois nós do modelo. Árvores são estruturas em que cada nó possui, no máximo, um outro nó predecessor na ordem causal. Já o termo poli-árvore é um tipo de grafo no qual não existem ciclos (ou *loops*), não importando as direções das arestas. Um grafo geral não se limita como os anteriores, podendo existir ciclos e diversos caminhos entre dois nós de uma rede. Grafos gerais representam as estruturas de redes Bayesianas Gerais.

- **Ordenação causal dos nós**: Muitos dos métodos para criação da estrutura via dados requer a ordenação causal dos nós do modelo. A ordenação determina a

precedência causal entre os nós (ou variáveis). Isso quer dizer que se uma variável A influencia B, então a variável A deve aparecer antes na lista de precedência de variáveis. Utilizou-se também a Figura 2.2 para exemplificar a ordenação causal dos nós de uma rede Bayesiana. Nela é visto que a variável A influencia B que influencia C. Nesse caso, a ordenação possível dessas variáveis é {A, B, C}. Quando a ordenação causal for necessária, mas não for passada como entrada para o método de busca de RBCs, o mesmo irá criar essa ordenação antes de poder moldar a estrutura da rede Bayesiana.

- **Tipo de algoritmo para criar a estrutura**: Considere as técnicas de Busca e Pontuação (*Search & Scoring*), Restrições, Híbridas e Programação Dinâmica. Técnicas baseadas em Busca e Pontuação realizam uma busca no espaço de estruturas, adicionando arestas à rede considerando que essas melhorem a qualidade da estrutura. A qualidade da estrutura é determinada pela métrica de pontuação ou *score*. Além dos diferentes tipos de métricas de pontuação, existem também diversos tipos de buscas. O segundo tipo de algoritmo é aquele baseado em Restrições. Para poder criar a estrutura, esse tipo de método analisa as dependências condicionais entre as variáveis para poder adicionar arestas ao modelo. Isso é geralmente feito por meio de testes de independência condicional (*Conditional Independence tests* ou CI-tests). Abordagens Híbridas unem os dois últimos tipos de técnicas para poder construir a estrutura da RB. Por fim, utilizar Programação Dinâmica é similar a realizar Busca e Pontuação, mas sem considerar o aspecto de busca.

Em um primeiro momento, métodos baseados em Programação Dinâmica não serão o foco deste trabalho. Diversos tipos de técnicas recentes são mencionadas por Daly et al. [2011], caso se queira aprofundar nesse tema. Métodos Híbridos terão um enfoque menor, como será visto adiante. Daly et al. [2011] também menciona diversos outros tipos de técnicas híbridas presentes na literatura. Contudo, a direção seguida por este trabalho está nas técnicas baseadas em Busca e Pontuação e em Restrições.

2.1.2.1 Algoritmos baseados em Busca e Pontuação

Existem diversos trabalhos que se baseiam na abordagem de Busca e Pontuação. Geralmente, este tipo de abordagem possui dois componentes principais: o componente de busca e o componente de pontuação (*score*).

De acordo com Daly et al. [2011], existe uma infinidade de métodos de busca: buscas gulosas, buscas *branch and bound*, algoritmos genéticos, *simulated annealing*,

particle swarm optimization (PSO), buscas híbridas e outros tipos de heurísticas específicas para busca.

Para os diferentes tipos de métricas ou funções de pontuação (*scoring metrics*), será seguida a separação feita por Cheng et al. [2002]. Mesmo que existam outras métricas, as apresentadas aqui distinguem bem e de forma geral essas funções. Os tipos são: Entropia, Bayesiano e Tamanho de Descrição Mínima (*Minimum Description Length - MDL*).

O primeiro trabalho relevante na literatura, o qual foi inspiração para diversos outros métodos, foi apresentado em Chow & Liu [1968]. Sua ideia está em um algoritmo capaz de aprender estruturas de redes Bayesianas na forma de árvores a partir de dados amostrais. Na proposta feita *Chow & Liu*, o método não requer a ordenação causal das variáveis, utiliza uma busca gulosa para adicionar arestas à rede (modificação do algoritmo de Kruskal [Cormen et al., 2009] para árvore geradora com peso máximo) e a métrica de pontuação utilizada é baseada na entropia da rede. Uma das principais motivações da utilização desse algoritmo está em sua complexidade, a qual está limitada em $O(n^2)$ cálculos de dependência em pares. Outra grande motivação está no fato do algoritmo escolher a melhor aproximação de uma distribuição de ordem N por meio do produto de N-1 distribuições de componentes de segunda ordem. É garantido, por essa aproximação, que o método maximiza a função de verossimilhança e, portanto, é um estimador de verossimilhança máximo da distribuição de dependência em árvore.

Rebane & Pearl [1987] continuaram o trabalho de *Chow & Liu*, estendendo a abordagem para poli-árvores. A diferença para o algoritmo de *Chow & Liu* está na capacidade de desenvolvimento de um modelo de dependência mais rico (informativo), ou seja, um modelo que pode conter uma maior quantidade de arestas direcionadas. O algoritmo proposto por *Rebane & Pearl* é apenas uma extensão do algoritmo de *Chow & Liu* e, portanto, o mesmo herdará todas as suas características, inclusive sua complexidade.

As demais técnicas que serão apresentadas a seguir serão formas para representar dados amostrais a partir de redes Bayesianas gerais, ou seja, a representação da estrutura consiste de um grafo direcionado.

Kulató, proposto por Herskovits & Cooper [1991], é um sistema que tem como entrada uma base de dados e a ordenação causal das variáveis, gerando como saída a estrutura da rede Bayesiana, que captura as relações de dependência daqueles dados. É um método baseado na entropia da rede para determinar seu *score* à medida que adiciona arestas no modelo (rede). Inicialmente, assume que todas as variáveis na base de dados são marginalmente independentes. Por meio de uma busca gulosa, adiciona incrementalmente arestas direcionadas entre pares de nós (ou variáveis), caso

se mantenha a aciclicidade e diminua a entropia geral da rede Bayesiana naquele passo. A condição de parada do algoritmo está determinada pela não existência de um nó pai (na lista de ordenação causal) que precede o nó atual capaz de diminuir sua entropia. Quando isso ocorre, o algoritmo para. O Kulató apresenta complexidade geral igual a $O(n^4 \times 2^n)$.

Cooper & Herskovits [1992] propuseram o K2, uma abordagem baseada no Kulató. A diferença entre Kulató e K2 está em sua métrica de *score*. Enquanto o primeiro utiliza uma medida baseada na entropia mínima da rede, o segundo se baseia em um *score* Bayesiano, em que se almeja maximizar a probabilidade da estrutura resultante. No artigo de *Cooper & Herskovits* também se explora outros métodos de busca, como o K2 Reverso (K2R). A distinção é que o K2R começa com a rede totalmente conectada (respeitando a ordenação causal), removendo nós de forma gulosa de acordo com a métrica Bayesiana. Outra ideia está em aplicar K2 e K2R, verificando qual é a estrutura gerada mais provável, e utilizá-la. A complexidade do K2 é igual a $O(m \times n^4 \times r)$, onde m é a quantidade de exemplos na base de dados e r o número máximo de possíveis valores para qualquer variável dessa base.

Continuando os dois últimos trabalhos, Peng & Ding [2003] desenvolveram o K2+, que como o próprio nome determina, é uma extensão do K2. Três melhorias sistemáticas foram realizadas:

1. Criou-se uma busca linear de nós pais para gerar um grafo candidato.

2. Desenvolveu-se uma abordagem completa/abrangente para eliminar ciclos utilizando a perda de verossimilhança mínima (*minimal likelihood loss*), uma heurística em que remoções de ciclos menores têm preferência a remoções de ciclos maiores. A justificativa para isso é que arestas presentes em ciclos menores também estão presentes em ciclos maiores, portanto a remoção de uma implica na remoção da outra. Nesse item, também pode-se reparar arestas cortadas, sempre que a remoção de uma aresta possa deixar o grafo candidato não ótimo.

3. Propôs-se também uma abordagem de pertubação de estrutura para avaliar a estabilidade da rede e um método de melhoria da estabilidade para refinar sua estrutura.

A complexidade da versão modificada do K2, o K2+ é $O(n^2)$ para construir o grafo candidato por completo.

Suzuki [1999] apresentou um algoritmo de aprendizado da estrutura de redes Bayesianas o qual usa o princípio do tamanho de descrição mínima ou *MDL principle* [Rissanen, 1978; Lam & Bacchus, 1994] . Para esse algoritmo é exigida a ordenação

causal das variáveis presentes na base de dados utilizada. O objetivo do MDL é selecionar as dependências entre variáveis (regras) que englobem tanto a simplicidade do modelo, quanto sua qualidade em relação aos exemplos (representação dos dados pelo modelo criado). Ponderando-se esse fato, aplica-se uma técnica de busca *Branch and Bound* [Land & Doig, 1960; Lawler & Wood, 1966] para encontrar uma estrutura da rede. Como o algoritmo de *Suzuki* minimiza o tamanho da descrição, ele eventualmente seleciona a estrutura verdadeira (ótima) da rede quando o tamanho das amostras tendem ao infinito, contudo ele é bastante ineficiente.

2.1.2.2 Algoritmos baseados em Restrições

Um algoritmo baseado em Restrições apresenta como componente principal algum teste de independência condicional. A ideia dessa classe de algoritmo é analisar/explorar os relacionamentos de dependência condicional entre as variáveis e utilizar esses relacionamentos como restrições para construir a rede Bayesiana. Para isso, faz-se uma análise a partir de um teste de independência condicional. Essas análises podem ser feitas com o teste χ^2 ou com o teste de informação mútua, por exemplo.

Srinivas et al. [1990] desenvolveram o algoritmo SRA (sigla para abreviação dos sobrenomes dos autores do artigo: Srinivas, Russel e Agogino). O SRA é uma abordagem para redes Bayesianas gerais e requer a ordenação causal dos nós para construção da estrutura da rede. O objetivo do SRA é construir automaticamente uma rede Bayesiana esparsa, que revela explicitamente o máximo de informações a respeito da independência condicional, de forma mais abrangente possível. O SRA molda a estrutura da rede incrementalmente, tentando manter o mínimo possível de arcos adicionados em cada passo, considerando que a informação de um especialista é utilizada para guiar uma busca heurística para essa finalidade. Um teste de independência estatístico é considerado um modelo de probabilidade que responde a requisições sobre independências de domínio. O número total de testes de independência condicional requerido pelo SRA é igual a $O(2^n)$, onde n é a quantidade de nós da rede. Quando se limita a p o número de pais (predecessores causais) que um nó pode ter, a complexidade passa a ser polinomial: $O(n^{p+2})$.

Spirtes & Glymour [1991] descrevem dois algoritmos: o SGS e o PC. Ambos são utilizados para redes Bayesianas gerais e não requerem a ordenação causal das variáveis. SGS é considerado bastante ineficiente, pois como cada par de variáveis requer testes envolvendo todo subconjunto de variáveis restantes, isso gera uma operação de natureza exponencial. O algoritmo PC é uma variação do SGS, que é mais rápido, possuindo uma complexidade polinomial: $O(n^{k+2})$, onde k é o grau máximo de qualquer nó na

estrutura verdadeira da rede. Contudo, o PC pode produzir erros na remoção de arco. Essas falhas são explicadas porque o PC testa somente a d-separação entre nós X e Y em um DAG, usando subconjuntos de vizinhos de X e Y.

Cheng et al. [1997a,b] abordam em seus trabalhos dois algoritmos para criação da estrutura de uma rede Bayesiana a partir de dados. Os dois algoritmos serão utilizados na estrutura funcional de alguns algoritmos da subseção 2.1.3 para desenvolvimento de classificadores específicos (Cheng & Greiner [1999] e Cheng & Greiner [2001]). Ambos os algoritmos se inspiram na teoria da informação, analisando as dependências condicionais entre os nós da rede a partir de um teste de informação mútua condicional para criar redes Bayesianas com múltiplas conexões.

Cheng et al. [1997a] considera um algoritmo que depende da ordenação causal dos atributos da base de dados para construção da rede. O algoritmo é composto basicamente por três fases: Esboço (*Drafting*), Expansão (*Thickening*) e Refinamento (*Thining*). Por meio dessas fases, o algoritmo consegue atingir uma complexidade igual a $O(n^2)$ em relação a testes de independência condicional, que são computacionalmente caros. As três fases são descritas nos itens a seguir:

1. **Esboço:** Computa a informação mútua de cada par de vértices como uma medida de proximidade e cria um esboço da rede baseando-se nessa informação.

2. **Expansão:** O algoritmo adiciona arestas quando os pares de nós não podem ser d-separados. O resultado dessa fase é um mapa de independência (I-Map) do modelo de dependência.

3. **Refinamento:** Cada arco do I-Map é examinado utilizando testes de independência condicional e serão removidos se dois vértices podem ser d-separados. O resultado dessa fase é um I-Map Mínimo.

O algoritmo presente no trabalho de Cheng et al. [1997b] é semelhante ao último apresentado porque também realiza as três fases descritas anteriormente. Sua diferença primordial é que este não releva a informação da ordenação causal dos atributos da base de dados como entrada para o algoritmo. Por essa razão, o algoritmo possui dois desafios: (1) determinar se dois nós da rede são condicionalmente independentes.; e (2) orientar as arestas no grafo aprendido. Com a incorporação de métodos que solucionam essas duas questões, o algoritmo passa a ter uma complexidade igual a $O(n^4)$ em termos de testes de independência condicional.

Para um estudo mais detalhado comparando as técnicas de Cheng et al. [1997a] e de Cheng et al. [1997b], o trabalho dos mesmos autores, Cheng et al. [1998], compara e determina diversos conceitos relevantes sobre os algoritmos.

Um ponto que merece ser destacado é que a fase de Esboço dos algoritmos anteriores é apenas uma modificação do método proposto por Chow & Liu [1968]. A diferença está no tipo de estrutura criada, que no caso da fase de Esboço geram-se redes com conexões múltiplas, diferente da estrutura de árvore criada pela abordagem de *Chow & Liu*.

Em Cheng et al. [2002] são apresentados quatro algoritmos. Os dois primeiros são o SLA-π (*Simple Learning Algorithm* - com ordenação causal como entrada) e o SLA (*Simple Learning Algorithm* - livre da ordenação causal das variáveis). O SLA-π e o SLA são versões simplificadas para os algoritmos de Cheng et al. [1997a] e Cheng et al. [1997b], respectivamente. Eles são compostos por duas fases apenas: Expansão (*Thickening*) e Refinamento (*Thining*). A complexidade do SLA-π e do SLA são semelhantes às complexidades de seus algoritmos não-simplificados. Já os outros outros dois algoritmos são o TPDA-π (*Three-Phase Dependency Analysis Algorithm* - com ordenação causal como entrada) e o TPDA (*Three-Phase Dependency Analysis Algorithm* - livre da ordenação causal das variáveis). O TPDA-π e o TPDA são, respectivamente, os algoritmos apresentados por Cheng et al. [1997a] e Cheng et al. [1997b].

2.1.2.3 Abordagens Híbridas

Existem diversos tipos de abordagens híbridas na literatura. Contudo, este trabalho se concentrará inicialmente em duas abordagens que serão descritas a seguir.

Como visto anteriormente, grande parte dos métodos relatados necessitam da ordenação causal das variáveis como entrada do algoritmo de construção da estrutura da rede Bayesiana. O algoritmo descrito por Singh & Valtorta [1995] diferencia-se dos demais por apresentar uma proposta híbrida. Primeiro, testes de independência condicional são utilizados para gerar uma ordenação causal das variáveis da base de dados, que é utilizada então para recuperar a estrutura da rede Bayesiana de acordo com um método que não se baseia em independência condicional. Assim, foi proposto durante esse trabalho o CB, uma abordagem para redes Bayesianas gerais que combina o PC (para ordenação causal das variáveis) e o K2 (para construir a estrutura da rede). O CB é limitado por uma complexidade exponencial, para o pior caso.

BENEDICT (*BElief NEtworks DIscovery using Cut-set Techniques*) foi proposto por Acid & de Campos [1996, 2001]. É um sistema que possui como entrada uma base de dados e a ordenação causal das variáveis, gerando como saída a estrutura da rede Bayesiana de crença. Sua ideia básica está na mensuração das discrepâncias entre as independências condicionais representadas em qualquer rede candidata e aquelas

apresentadas pela base de dados. A rede candidata com a menor dessas discrepâncias é a melhor rede que modela os dados. Para realizar esse processo, é feita uma busca heurística que utiliza o conceito de d-separação (restrições de dependência condicional), e para medir a discrepância da rede candidata usa-se o conceito de entropia cruzada de Kullback-Leibler [Kullback & Leibler, 1951]. Esse tipo de entropia mede o grau de dependência entre X e Y, dado que se conhece Z. A medida geral de discrepância, que deve ser minimizada, tem como função ponderar qual das redes candidatas é a melhor e essa medida é definida como a soma de todos os graus de dependência para pares de nós não-adjacentes (entropia cruzada), dados seus conjuntos mínimos de d-separação. Pondera-se que a complexidade do BENEDICT no pior caso é $O(n^6)$.

2.1.3 Aprendizado de Redes Bayesianas para Classificação

Esta subseção apresenta os diversos tipos de classificadores de redes Bayesianas que estão presentes na literatura. A diferença dos algoritmos que serão discutidos nesta subseção em relação à anterior está na existência do atributo classe, já que aqui será tratado de classificadores.

Assim, as conexões entre atributos para classificadores são diferenciadas, dependendo em grande parte do atributo classe. O algoritmo de aprendizado de RBCs moldará as relações entre os n atributos da base $(A_1, ..., A_n)$ e também as relações causais entre este conjunto de atributos A e o atributo classe C. Algoritmos de aprendizado da estrutura de redes Bayesianas gerais não consideram como parte do processo um nó especial como o nó classe. Sendo assim, algoritmos de RBs gerais apenas podem ser modificados para poder gerar um classificador não-específico.

O que torna esses classificadores específicos importantes frente aos demais (RNAs, KNNs e Árvores de Decisão) é a questão da geração de um modelo adequado de classificação, visto que grande parte deles leva em consideração as dependências (influências) entre pares de atributos.

De acordo com Cheng & Greiner [1999], redes Bayesianas não eram consideradas classificadores até o surgimento do Naïve Bayes (NB), um tipo simples de RBs que assume que os atributos da base são independentes dado o nó de classificação. Uma descrição completa sobre o NB está presente em Langley et al. [1992], onde é visto que esse classificador possui resultados surpreendentemente efetivos frente a outros classificadores sofisticados. Como o NB considera a independência entre os atributos, não existe um processo de aprendizado da estrutura da RB, visto que essa é fixa. Apenas os parâmetros (Tabela de Probabilidade Condicional) e a sua saída são aprendidos.

Em Friedman et al. [1997], o classificador Tree-Augmented Naïve Bayes (TAN) é

apresentado. Ele supõe que a variável classe não possui predecessores causais ("pais"), e que cada atributo deve ter como predecessor a variável classe e, no máximo, um outro atributo. Tanto no trabalho de *Friedman et al.* quanto nos trabalhos de *Cheng & Greiner* (descritos posteriormente) são utilizadas modificações do trabalho de Chow & Liu [1968] para mapear as relações causais entre atributos de uma base de dados.

O trabalho de *Friedman et al.* também apresenta uma discussão a respeito de classificadores Bayesianos não-restritivos, ou seja, aqueles que assumem a estrutura de uma rede Bayesiana geral. Para isso, foi desenvolvido nesse trabalho um classificador baseado na minimização da métrica de pontuação MDL. Foi comprovado, via resultados experimentais, que classificadores Bayesianos não-restritivos podem melhorar a precisão dos resultados em bases de dados que contêm fortes iterações (relações) entre atributos. Também é apontado no texto sobre a importância da escolha de uma métrica de pontuação, visto os problemas encontrados na métrica MDL: não necessariamente otimiza a precisão de classificação das redes aprendidas.

Na sequência de trabalhos, em Cheng & Greiner [1999] e Cheng & Greiner [2001] são definidos cinco algoritmos para aprendizado de classificadores Bayesianos. Dois deles são o NB e o TAN. Os três algoritmos restantes são descritos nos três próximos parágrafos.

O *Bayesian network Augmented Naïve Bayes* (BAN) estende o classificador TAN permitindo que os atributos formem um grafo arbitrário em vez de somente uma árvore. Aprender tal tipo de estrutura é menos eficiente. Friedman et al. [1997] apresenta uma métrica baseada no conceito de MDL para aprender um BAN. Já Cheng & Greiner [1999] estudam um algoritmo distinto baseado em testes de independência condicional (restrições de dependência da RB).

Já a *Bayesian Multi-net* (BM) pode ser vista como uma generalização do BAN. Para isso, a BM distingue como as relações entre nós características são criadas. Os nós características representam os atributos previsores da base de dados. Desse modo, enquanto o BAN força os relacionamentos entre os nós características serem os mesmos para todos os valores possíveis do nó classe, a BM permite que as relações entre os nós características sejam diferentes, ou seja, para diferentes valores do nó classe, os nós características podem formar uma rede local de diferentes estruturas. Mesmo que existam várias redes, BM é menos complexa que BAN devido à simplicidade da rede criada, pois o BAN precisa ter uma estrutura complexa para ditar todos os relacionamentos entre atributos.

Por último, o *General Bayesian Network* (GBN) é um tipo de classificador de BN não-restritivo que considera, diferente dos últimos classificadores apresentados, o nó classe como um nó comum. Assim, todos os nós da RB utilizada para classificação

são equivalentes.

Em Santos et al. [2011] é proposta uma abordagem baseada nos conceitos de cobertura de Markov, bem como no de cobertura aproximada de Markov (ver Subseção 2.1.1). Ambos conceitos são utilizados para reduzir o número de nós que forma a rede Bayesiana a ser induzida de dados. Dois algoritmos de classificação foram propostos no artigo referenciado: DMBC (Dynamic Markov Blanket Classifier) e A-DMBC (Approximate DMBC). Como discutido durante o texto, ambos os algoritmos foram baseados no K2, que pode ser desenvolvido especificadamente para problemas de classificação. O DMBC (A-DMBC) utiliza a cobertura (aproximada) de Markov do nó classe para poder eliminar atributos não relevantes para classificação, gerando classificadores bem específicos. Os resultados obtidos por ambos os algoritmos foram competitivos com o NB e com o TAN, mas não foram melhores que o K2. Contudo, percebeu-se pelos resultados do DMBC e A-DMBC que esses conseguem prover boas acurácias de classificação e também diminuir o custo computacional em relação ao K2 para aprender o classificador.

Em Sacha [1999b] e Sacha et al. [2002] é apresentada outra família de classificadores de Redes Bayesianas. As ideias propostas por ambos os trabalhos é interessante pois considera que a combinação de primitivas de algoritmos (ou operadores dos algoritmos) podem ser combinadas e modificadas para produzir diversos tipos de algoritmos de aprendizado de redes Bayesianas. Essas primitivas são, por exemplo, técnicas de busca do algoritmo ou sua ponderação (métrica de pontuação) para avaliar a estrutura da rede candidata.

Essa família é composta por cinco classificadores, que são brevemente descritos nos itens que seguem:

1. **Selective Tree-Augmented Naïve Bayes (STAN):** Um algoritmo de busca que decide quais nós atributos dependem do nó classe. Utiliza uma busca gulosa para adicionar arestas à RB, assumindo inicialmente um conjunto vazio de "filhos" de cada nó. A cada passo adiciona um novo "filho" a um outro nó, caso esse "filho" gere um melhor resultado da métrica de pontuação utilizada. Como o TAN, uma árvore é a estrutura que representa as dependências entre os atributos.

2. **Selective Tree-Augmented Naïve Bayes with Discarding (STAND):** É similar ao STAN, fazendo uso de uma algoritmo de busca para decidir quais nós atributos dependem do nó classe. A diferença é que o STAND descarta atributos que não são ditos dependentes da variável classe. Em outras palavras, aplica uma seleção de atributos que determina quais são insignificantes para o objetivo da classificação, removendo-os da rede Bayesiana de classificação.

3. **Forest-Augmented Naïve Bayes (FAN):** Assume que todos os nós atributos dependem do nó classe. As dependências entre os atributos tem a forma de uma floresta (um número de árvores). A expansão da floresta é feita com base no algoritmo de Kruskal modificado para árvore geradora de peso máximo. O algoritmo original de Kruskal é descrito em Cormen et al. [2009]. Em alguns casos, a expansão da floresta pode ser vazia, produzindo a estrutura de um Naïve Bayes; ou também pode ter somente uma árvore geradora, sendo assim o classificador terá a estrutura de um TAN.

4. **Selective Forest-Augmented Naïve Bayes (SFAN):** Algoritmo semelhante ao FAN, diferenciando na utilização de um algoritmo de busca que decide quais nós atributos dependem do nó classe. É importante mencionar que a floresta criada pode abranger todos os nós atributos.

5. **Selective Forest-Augmented Naïve Bayes (SFAND):** Um algoritmo de busca que decide quais atributos dependem do nó classe, enquanto outros nós são descartados da rede Bayesiana caso esses não influenciem no processo de classificação. A floresta criada pode conter somente atributos que dependam da variável classe.

Cada um dos algoritmos descritos nos itens diferem em termos de um dilema entre complexidade computacional e qualidade da estrutura de rede criada. A Figura 2.6 ilustra a riqueza em termos das estruturas (diferentes tipos de relações causais) que podem ser produzidas por tais algoritmos, e suas relações com os classificadores Naïve Bayes e TAN.

2.2 Um sumário dos algoritmos de RBC

Esta seção apresenta um resumo dos principais algoritmos de RBC que serão utilizados neste trabalho, como eles se interrelacionam e seus principais componentes. Esses componentes são essenciais durante a evolução dos algoritmos no Capítulo 3. Os algoritmos são representados utilizando uma gramática, e descritos na Figura 2.7.

Inicialmente, o *Aprendizado de Redes Bayesianas de Classificação* é dividido em dois componentes principais: *Aprendizado da Estrutura* e *Aprendizado dos Parâmetros*. O *Aprendizado da Estrutura* pode ser feito tanto por métodos de busca específicos para RBCs (K2, TAN, etc) quanto por métodos de busca tradicionais (*Hill Climbing*, *Simulated Annealing*, etc). Já o *Aprendizado dos Parâmetros* é feito após a criação da estrutura, onde se realizam estimativas a partir dos dados, dada a estrutura da rede.

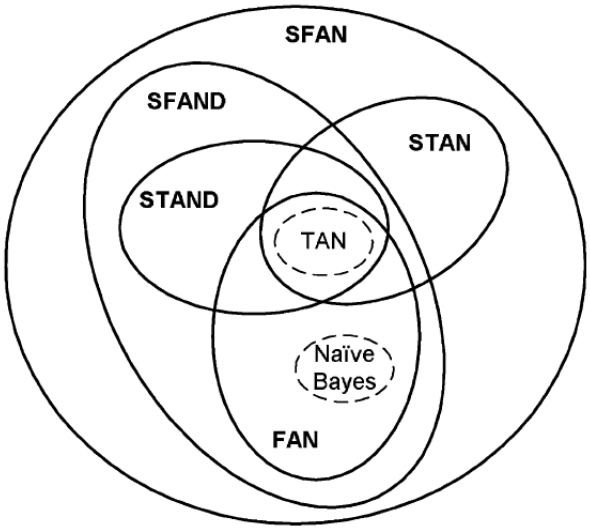


Figura 2.6: Família de classificadores apresentada por Sacha [1999b]; Sacha et al. [2002] e sua relação com o Naïve Bayes e com o TAN. Linhas tracejadas indicam classificadores estado-da-arte e linhas sólidas representam classificadores propostos.

O foco deste trabalho é na fase de *Aprendizado da Estrutura*, onde existem dois caminhos possíveis, um *Restrito* e outro *Flexível*. No caminho restrito, o único algoritmo presente é o Naïve Bayes, o qual possui uma busca de estrutura fixa. Já para o outro caminho, que gera estruturas mais flexíveis quanto às relações, três subdivisões de tipos de algoritmos são delimitadas: *Busca com Pontuação*, *Busca por Restrições* ou *Híbridas*. Além disso, ao final dessas categorias de busca, uma cobertura de Markov é aplicada em relação à variável classe.

Existem diversas outras subdivisões na gramática que definem os principais componentes de RBCs. Contudo, ao final pode-se gerar basicamente os seguintes algoritmos de aprendizado de estrutura de RBC: *Tree Augmented Naïve Bayes (TAN) using Conditional Independence (CI) Tests*, *Inductive Causation Search* (ICS) [Verma & Pearl, 1992], *General Augmented Naïve Bayes* (GAN), Busca Gulosa (BG) realizada pelo K2 (derivado do Kulató) e suas variações, *Hill Climbing* (HC) [Hesar et al., 2012], *Look Ahead in Good Directions Hill Climbing* (LAGD HC) [Abramovici et al., 2008], *Repeated Hill Climbing* (RHC) [Hesar et al., 2012], Busca Tabu (Tabu) [Bouckaert, 1995] e *Simulated Annealing* (SA) [Bouckaert, 1995].

As métricas de pontuação locais e globais utilizadas para a categoria *Pontuação* listadas estão detalhadas nos trabalhos de Bouckaert [1995], Bouckaert et al. [2013], Witten et al. [2011] e Sacha [1999b].

Aprendizado de RBCs ← Aprender a Estrutura de RBCs **E** Aprender os seus Parâmetros
Aprender a Estrutura de RBCs ← Restrita **OU** Flexível
Restrita ← Naïve Bayes
Naïve Bayes ← Criar arestas partindo da classe para os atributos previsores.
Flexível ← (Busca com Pontuação **OU** Busca por Restrições **OU** Híbridas) **OU** Cobertura de Markov
Busca com Pontuação ← Filtro de Atributos **E** (Busca **E** Pontuação)
Filtro de Atributos ← Nulo **OU** (Seleção **OU** Descarte **COM** Pontuação)
Nulo ← Não aplicar filtro algum e manter todos os atributos dependentes à classe
Seleção ← Identificar atributos não-dependentes à classe **E**
Não adicionar arestas entre atributos não-dependentes e classe **E**
Manter atributos no restante da busca
Descarte ← Identificar atributos não-dependentes à classe **E**
Não adicionar arestas entre atributos não-dependentes e classe **E**
Remover atributos do restante da busca
Busca ← Busca com Ordenação **OU** Busca sem Ordenação
Busca com Ordenação ← Buscas Gulosas com Ordem **E** Parâmetros Gulosos
Parâmetros Gulosos ← Estrutura NB **E** Número de Pais **E** Inversão de Arcos
Estrutura NB ← Definir estrutura do NB como inicial **OU** Definir estrutura nula como inicial
Número de Pais ← Escolher a quantidade de pais máxima de um nó da RBC no intervalo inteiro $[1, 10]$
Inversão de Arcos ← Inversão de arcos permitida na busca **OU** Não permitir tal operação na busca
Buscas Gulosas com Ordem ← BG K2 **OU** BG K2 Reverse **OU** BG K2+ **OU** Melhor das três
Busca sem Ordenação ← (Buscas Gulosas sem Ordem **E** Parâmetros Gulosos) **OU** Heurísticas de Otimização **OU** Abordagem Específica
Buscas Gulosas sem Ordem ← HC **OU** LAGD HC **OU** RHC
HC ← Aplicar busca gulosa para adicionar, remover e inverter arestas.
LAGD HC ← Considerar estruturas vizinhas até o Tamanho do Passo **E**
Verificar o Número de Operações para o cálculo da melhor sequência de operações.
Tamanho do Passo ← Escolher o valor inteiro do tamanho do passo no intervalo $[1, 5]$
Número de Operações ← Escolher o valor inteiro do número de operações no intervalo $[1, 5]$
RHC ← Perturbar aleatoriamente a estrutura criada pelo HC **COM** um número de Execuções definido
Heurísticas de Otimização ← (Busca Tabu **E** Parâmetros Gulosos) **OU** SA
Busca Tabu ← Criar uma estrutura de RBC arbitrária **E** Melhorá-la recursivamente em um número de Execuções **COM** o uso de uma Lista Tabu
Lista Tabu ← Determinar o seu tamanho da lista dado o intervalo inteiro $[1, 10]$
Execuções ← Definir um número baixo de execuções considerando o intervalo inteiro $[1, 10]$
SA ← Criar uma estrutura que possui Temperatura Inicial alta **E** Estabilizar essa solução a partir de um número de Iterações **E** uma taxa δ de atualização
Iterações ← Definir o valor do número de iterações pelo intervalo inteiro $[10, 250]$
Temperatura Inicial ← Determinar o valor de temperatura inicial pelo intervalo real $[10.0, 300.0]$
Delta (δ) ← Determinar o valor da taxa de atualização δ no intervalo real $[0.0, 1.0]$
Abordagem Personalizada ← GAN
GAN ← Gerar arestas da árvore **COM** Pontuação **E** Criar estrutura da RBC **COM** uma Complexidade Máxima
Complexidade Máxima ← Árvore **OU** Floresta
Pontuação ← Métricas Locais **OU** Métricas Globais **OU** (Métricas Globais **E** Probabilidade da Classe)
Métricas Locais ← Bayesiana **OU** BDeu **OU** MDL **OU** AIC **OU** Entropia
Métricas Globais ← LOO-CV **OU** k-Fold-CV **OU** Cumulative-CV **OU** HGC **OU** SB **OU** LC **OU** LogC **OU** LLOO **OU** LKCV
Probabilidade da classe ← Retornar a probabilidade da classe como estimativa da acurácia **OU** Retornar o resultado de uma função *zero-one loss* para estimar a acurácia
Cobertura de Markov ← Aplicar cobertura de Markov em relação à variável classe **E** Descartar atributos fora da cobertura
Restrições ← TAN **OU** (Filtro de Atributos **E** ICS)
TAN ← Gerar arestas da árvore utilizando Testes de Independência Condicional **E** Criar estrutura
ICS ← Buscar o esqueleto da RBC com Testes de Independência Condicional **COM** Cardinalidade Máxima definida **E** Direcionar arestas no esqueleto para criar RBC
Cardinalidade Máxima ← Escolher a cardinalidade máxima da busca no intervalo inteiro $[1, 10]$
Híbridas ← GAN (H) **OU** ICS (H)
GAN (H) ← Gerar as arestas da árvore por um Teste de Independência Condicional **E** Continuar a busca original.
ICS (H) ← Filtro de Atributos **E** ICS
Aprender os seus Parâmetros ← Aprenda as tabelas de probabilidades segundo os dados e a estrutura

Figura 2.7: Um resumo dos Algoritmos para Redes Bayesianas de Classificação.

2.3 Algoritmos Evolucionários

Um algoritmo evolucionário simula o processo da biologia evolutiva baseado em hereditariedade, mutação, seleção natural e recombinação (ou cruzamento) para tentar encontrar soluções aproximadas em problemas de otimização ou de busca [Back & Schwefel, 1996; Eiben & Smith, 2003]. Nesses algoritmos, o conceito de população é utilizado para representar o espaço de soluções do problema, sendo essa população variante ao longo das iterações do algoritmo. Cada indivíduo dessa população é uma solução candidata para um dado problema. Esse indivíduo pode ser representado de diversas formas: uma árvore de derivação, um arranjo binário, uma arranjo inteiro, etc. Isso dependerá do problema que está sendo estudado e qual método pretende-se aplicar.

Segundo Back & Schwefel [1996], qualquer algoritmo evolucionário possui basicamente os mesmos passos, que são:

1. **Inicialização** - Gera-se uma população inicial aleatória com N indivíduos.

2. **Avaliação** ou **Fitness** - Cada indivíduo é avaliado de acordo com uma função de avaliação $F(x)$ que determina o quão apto ele é para resolver o problema em questão.

3. **Seleção** - Os indivíduos dessa população inicial são selecionados utilizando métodos que realizem implicitamente ou explicitamente um disputa entre indivíduos dessa população. A ideia é que ao final exista uma maior probabilidade dos mais aptos serem selecionados. Existem vários métodos de seleção, e aqui utilizamos a seleção por torneio. Na seleção por torneio, k indivíduos são selecionados aleatoriamente da população, e aquele com maior valor de *fitness* é considerado o vencedor. O indivíduo vencedor estará presente na população da próxima geração.

4. **Cruzamento** - Operador genético que faz com que os indivíduos selecionados sejam recombinados para gerar descendentes seguindo seu material genético. Para se recombinar dois indivíduos, segue-se uma probabilidade (taxa) de cruzamento Pc, que determina a porcentagem de vezes que o material genético desses dois indivíduos da população será recombinado a fim de produzir dois indivíduos filhos (descendentes) com material de ambos os indivíduos originais. Geralmente, probabilidades de cruzamento possuem valores altos. O tipo de cruzamento adotado foi o uniforme, em que dois indivíduos são escolhidos pelo processo de seleção (item anterior) e trocas entre o material genético desses dois indivíduos são feitas

de acordo com uma função de probabilidade uniforme. Caso a função retorne valores menores ou iguais a 0,5 para uma região dos indivíduos, os seus materiais genéticos são trocados. Caso contrário, os materiais genéticos dos indivíduos são mantidos.

5. **Mutação** - Também considerado um operador genético. Determina que os indivíduos que foram selecionados sofram mutação em regiões específicas do material genético do indivíduo. É utilizada com o objetivo de incluir novidades na população da próxima geração. Neste trabalho foi empregada a mutação uniforme, em que partes do indivíduo a serem modificadas são escolhidas de forma aleatória, seguindo uma função de probabilidade uniforme. Além disso, cada parte do indivíduo é modificada de acordo com suas restrições, ou seja, nenhuma novidade é incluída caso isso não faça sentido. Além disso, semelhante ao cruzamento, a mutação é aplicada de acordo com uma probabilidade Pm, que normalmente assume um valor baixo.

6. **Atualização da População** - Novos N indivíduos descendentes são gerados nos passos 4 e 5. Cria-se então uma nova população. Retorna-se ao passo 3 caso essa nova população não tenha alcançado o estado de convergência (critério de parada).

Esses passos são realizados até que a população gerada chegue ao critério de parada, que pode ser um máximo número de iterações T previamente especificado ou uma medida de erro mínimo, por exemplo. Ao final do processo evolucionário, o melhor indivíduo da população final, de acordo com a função de avaliação, é retornado como o indivíduo a ser utilizado como solução para o problema em questão.

A Figura 2.8 apresenta um esquema mostrando todos os passos de execução de um algoritmo evolucionário, os quais foram discutidos anteriormente. É importante frisar que a execução do algoritmo evolucionário exige a definição de um conjunto de parâmetros, que inclui o tamanho da população, taxas de cruzamento e mutação, e número de gerações (iterações), quando essa é utilizada como critério de parada.

Raidl [2005] menciona que existem diferentes tipos de algoritmos evolucionários, dentre eles: Algoritmos Genéticos, Programação Genética, Estratégias Evolucionárias e Programação Evolucionária. As principais diferenças entre essas abordagens estão na forma que soluções candidatas são representadas e como a seleção, recombinação e mutação são implementadas.

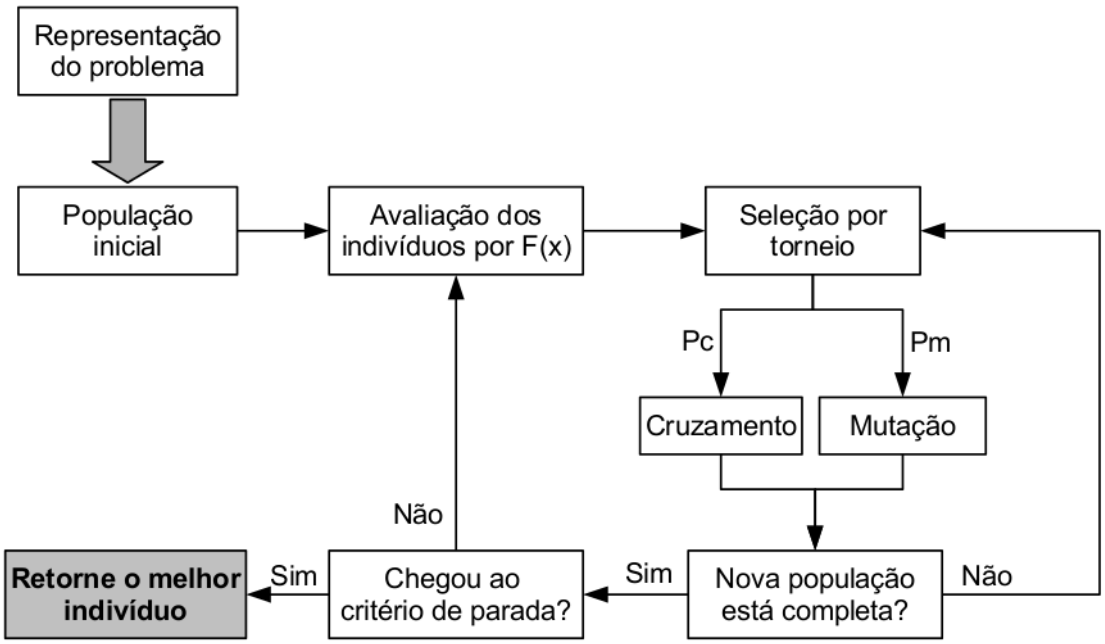


Figura 2.8: Passos básicos de um algoritmo evolucionário.

2.4 Trabalhos Relacionados

Existem basicamente na literatura três categorias de modelos de classificação que atualmente suportam evolução de algoritmos: Árvores de Decisão [Barros et al., 2012, 2013a,b], Redes Neurais Artificiais (RNAs)[Yao, 1999; Stanley & Miikkulainen, 2002] e Algoritmos de Indução de Regras [Pappa & Freitas, 2009a,b].

Na primeira classe de evolução de modelos de classificação estão os algoritmos de árvore de decisão, evoluídos pelo sistema proposto por Barros et al. [2012]. Os autores desenvolveram um algoritmo evolucionário para projetar automaticamente algoritmos de indução de árvores de decisão, chamado de HEAD-DT (Hyper-heuristic Evolutionary Algorithm for Automatically Designing Decision-Tree algorithms). No trabalho de *Barros et al.* foram identificados primeiramente as categorias de componentes dos algoritmos de árvores de decisão, que são: (i) critérios de ramificação; (ii) critérios de parada do algoritmo; (iii) regras para lidar com falta de valores; (iv) métodos de poda. Feito isso, um algoritmo evolucionário (AE) generativo foi aplicado, considerando que cada indivíduo nada mais é que um vetor inteiro de nove posições identificadas por essas quatro categorias de componentes, considerando que cada posição do indivíduo representa um tipo de componente dos algoritmos de indução de árvore de decisão.

No trabalho de Barros et al. [2012] foram consideradas nos experimentos apenas bases de dados (aplicações) de propósito geral. Já nos trabalhos mais recentes dos autores[Barros et al., 2013a,b] foram também analisadas aplicações de domínio real e,

além disso, situações em que existem diversas bases de treinamento (meta-treino) e teste (meta-teste). A partir desses novos cenários de teste, pôde-se realmente concluir que HEAD-DT é consistentemente o melhor método frente aos algoritmos estado-da-arte para indução de árvores de decisão.

Considerando outros tipos de modelos de classificação, pesquisadores na área de redes neurais têm investido muito esforço em técnicas de seleção e construção de componentes de tais algoritmos. Como discutido por Yao [1999], redes neurais podem ser evoluídas em três níveis distintos: escolha dos pesos das sinapses, modelagem da topologia e seleção da regra de aprendizado. Aqui são descritos os principais aspectos da *neuro-evolução*, definido em Floreano et al. [2008] como o processo de evolução de redes neurais artificiais a partir de algoritmos evolucionários. Inicialmente, a *neuro-evolução* seguia principalmente uma abordagem de seleção de algoritmos. Porém, atualmente, essa área está se deslocando para a direção de geração de algoritmos, como mencionado por Cantu-Paz & Kamath [2005]. O primeiro uso de termo *neuro-evolução* foi para evoluir os pesos de conexões de uma população de RNAs. Essa ideia básica cresceu e pesquisadores começaram a evoluir, juntamente com os pesos, a topologia da rede. Na última década, a busca por algoritmos que descrevem as regras de aprendizado tem avançado significadamente.

Um exemplo específico da *neuro-evolução* está no NEAT (*Neuroevolution of Augmenting Topologies*), criado por Stanley & Miikkulainen [2002]. Esse sistema é bem adequado para reforçar tarefas de aprendizado e evoluir topologias juntamente com os pesos.

Em Pappa & Freitas [2009a,b], os autores propõem um algoritmo evolucionário baseado em gramáticas para evoluir algoritmos de indução de regras. Uma gramática foi usada durante o processo de evolução para guiar a busca da programação genética. O processo de automaticamente evoluir algoritmos primeiro requer um estudo dos algoritmos projetados manualmente. No caso de Pappa & Freitas [2009b], a literatura se preocupa com a revisão de trabalhos de algoritmos de indução de regras e o seu conjunto de componentes foi identificado: busca da regra, avaliação e poda. Diferentes métodos de implementação desses componentes foram encontrados e adicionados à gramática. Finalmente, laços e declarações condicionais foram também adicionados, terminando com uma gramática com 26 regras de produção.

Baseada nessa gramática, um algoritmo de programação genética foi utilizado para gerar, avaliar e evoluir os algoritmos de indução de regras (que são os indivíduos). Como o algoritmo foi projetado pra trabalhar com qualquer tipo de base de dados, o desafio foi implementar uma função de avaliação (função de *fitness*) que garantisse a generalização dos algoritmos produzidos. Esse problema foi resolvido usando um

conjunto de bases de dados para calcular a qualidade (*fitness*) do indivíduo, nomeando-o como conjunto de meta-treino (*meta-training*). Portanto, para cada indivíduo da população, seu correspondente algoritmo de indução de regras foi traduzido em código Java e executado sobre o conjunto de meta-treino. Para cada base de dados no conjunto de meta-treino, um modelo foi gerado e sua acurácia no conjunto de teste foi calculada. A média de acurácia em todas as bases de dados no conjunto de meta-treino foi usada como a *fitness* do indivíduo.

Os resultados mostraram que programação genética pode gerar algoritmos de indução de regras diferentes daqueles já propostos na literatura e com acurácia de classificação competitiva. Seguindo outra abordagem de Pappa & Freitas [2009a], os autores também propuseram o uso do algoritmo para gerar algoritmos direcionados a bases de dados específicas ou com características similares.

Capítulo 3

Evolução de Algoritmos Redes Bayesianas de Classificação

O uso de modelos de classificação para mineração de dados tornou-se um campo de pesquisa bastante atrativo nos últimos anos, principalmente para assuntos relacionados à tomada de decisão e previsão. Nesse contexto, métodos estatísticos aparecem como uma alternativa interessante frente às demais abordagens pela inclusão do tratamento de incertezas em relação a elementos do mundo real. RBCs são exemplos desse tipo de método e os modelos gerados por tal técnica são visualmente compreensíveis e interpretáveis por um especialista de domínio. Por isso, existe um grande motivação para o estudo de tais algoritmos.

É importante frisar que mesmo possuindo uma compreensão e interpretação visual, RBCs exigem um maior esforço nesse quesito em relação aos modelos gerados por algoritmos de indução de árvores de decisão ou algoritmos de indução de regras, pois nelas estão inclusas conceitos de probabilidade (condicional), (in)dependência condicional, fluxo de influências probabilísticas, dentre outros.

Este trabalho propõe um algoritmo evolucionário (AE) para gerar automaticamente uma RBC utilizando conhecimento prévio de algoritmos construídos por projetistas humanos. Ressalta-se que a proposta central deste trabalho foi anteriormente publicada em Sá & Pappa [2013].

A ideia de construção de algoritmos personalizados/adaptados que gerem o modelo de classificação mais adequado aos dados vai ao encontro com a literatura de meta-aprendizado, principalmente com a parte de geração de algoritmos por seleção e construção, como discutido no Capítulo 1.

Como mencionado anteriormente, o processo de aprendizado em redes Bayesianas consiste basicamente de dois passos: aprendizado da estrutura e aprendizado dos

parâmetros. De acordo com Salama & Freitas [2014], aprender os parâmetros de uma RB é relativamente simples e direto, dado que se tenha a estrutura da rede definida com dependências específicas entre as variáveis. Por essa razão e por limitações no *framework* de aprendizado de máquina utilizado, o presente trabalho se concentrará no aprendizado da estrutura da RBC, considerando que os parâmetros da rede não sejam tão relevantes dentro do espaço de busca dos componentes de RBCs.

Este capítulo apresenta uma visão detalhada dos principais elementos utilizados durante a evolução dos algoritmos de classificação Bayesianos. A Seção 3.1 descreve de forma geral como o método funciona, enquanto a Seção 3.2 discute as partes essenciais dos algoritmos de RBCs, que são os componentes de tais algoritmos. A Seção 3.3 define as diferentes formas utilizadas para representar indivíduos. A representação do indivíduo aqui empregada considera em grande parte os principais componentes comuns a diferentes algoritmos de RBCs, sumarizados na Seção 2.2. A Seção 3.4 se concentra na função de avaliação (*fitness*), definindo-a apropriadamente. A Seção 3.6 mostra uma adaptação do algoritmo genético descrito na Seção 3.1 para trabalhar com métricas de complexidade de bases de dados. O intuito do uso de tais métricas está relacionado ao uso do AG para grupos de bases de dados, em que as métricas são utilizadas para guiá-lo a fim de encontrar soluções semelhantes para tipos de dados com as mesmas particularidades.

3.1 Visão Geral

Esta seção apresenta a abordagem proposta para evoluir automaticamente algoritmos de RBCs. Como outros trabalhos já propostos na literatura, a ideia do presente trabalho é gerar RBCs personalizadas/adaptadas a bases de dados específicas ou grupos de bases de dados com características similares, ao invés de simplesmente executar experimentos para selecionar um entre vários métodos e testar diferentes configurações de parâmetros.

A abordagem proposta possui dois componentes principais: (1) um conjunto de componentes de RBCs relacionados; (2) um método de busca para explorar as diferentes combinações desses componentes. A Figura 3.1 (já apresentada no Capítulo 1) mostra o processo de funcionamento seguido pelo método. O mesmo recebe como entrada um conjunto de bases de dados e, considerando todos os componentes das RBCs identificados e o método de busca, retorna uma RBC (ou várias) personalizada(s) às particularidades dos dados. A RBC retornada pelo método de busca será adequada ao domínio de dados de entrada, mas isso não significa que não possa generalizar para

outras bases, como será mostrado no Capítulo 4.

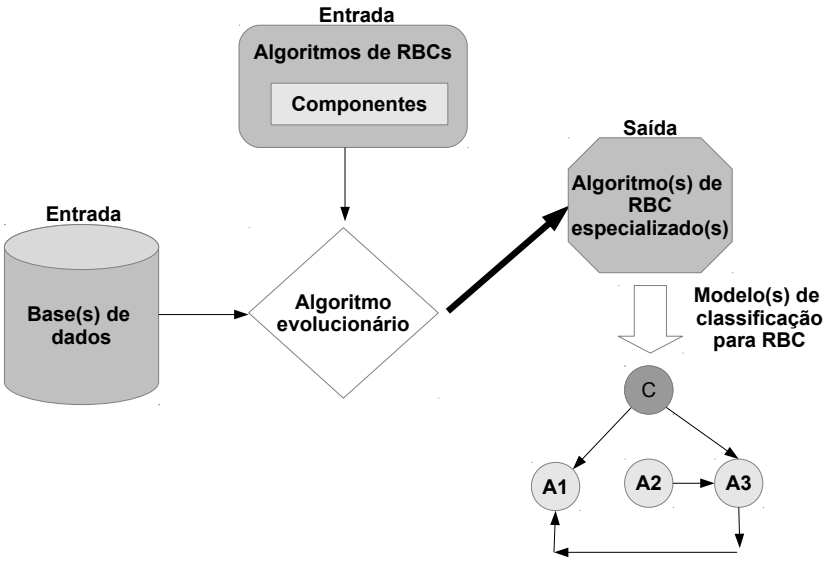


Figura 3.1: Processo de evolução automática de algoritmos de RBCs.

Ressalta-se também que o conjunto de bases de dados utilizadas como entrada pode conter tanto uma base de dados de treinamento e outra de teste, quanto grupos de bases de dados de treino e grupos distintos de teste.

O método de busca, segundo componente desta abordagem, é um Algoritmo Genético (AG) com codificação real para explorar o espaço de RBCs. No AG, cada indivíduo representa um algoritmo de RBC (mais detalhes na Seção 3.3),e na primeira geração os indivíduos são aleatoriamente gerados a partir de uma combinação dos componentes disponíveis. Durante a avaliação dos indivíduos, um mapeamento entre o indivíduo e um algoritmo de RBC é realizado (mais detalhes na Seção 3.4) .

Seguindo o processo de evolução, os indivíduos são submetidos aos operadores de cruzamento uniforme e mutação de um ponto para gerar uma nova população. Neste trabalho, também foi feito o uso do elitismo, garantindo que que o melhor indivíduo da população anterior estará presente na atual. Depois de um número pré-definido de gerações, o melhor algoritmo de RBC encontrado pelo AG é retornado. Assim, esse algoritmo de RBC é executado em um conjunto de teste, podendo considerar um domínio de aplicação semelhante ou totalmente distinto.

Os principais elementos relacionados ao AG são a *representação do indivíduo*, que depende fortemente dos *componentes dos algoritmos de RBCs*, e a *função de avaliação dos indivíduos (*fitness*)*. Esses elementos serão discutidos nas três próximas seções.

3.2 Componentes dos Algoritmos de RBCs

Um dos passos mais importantes desta pesquisa é identificar uma lista de componentes relevantes e suas dependências para que o algoritmo de busca (neste caso, um AG) possa explorar o espaço de combinação desses componentes e construir uma solução (algoritmo de RBC) personalizada a uma base de dados específica ou a conjuntos de bases de dados.

A Figura 3.2 apresenta um esboço do processo de aprendizado em RBCs, o qual é dividido em duas partes principais: o aprendizado da estrutura e o aprendizado dos parâmetros. O principal componente do aprendizado da estrutura é o método de busca, que pode assumir 12 diferentes opções, conforme a Figura 3.2.

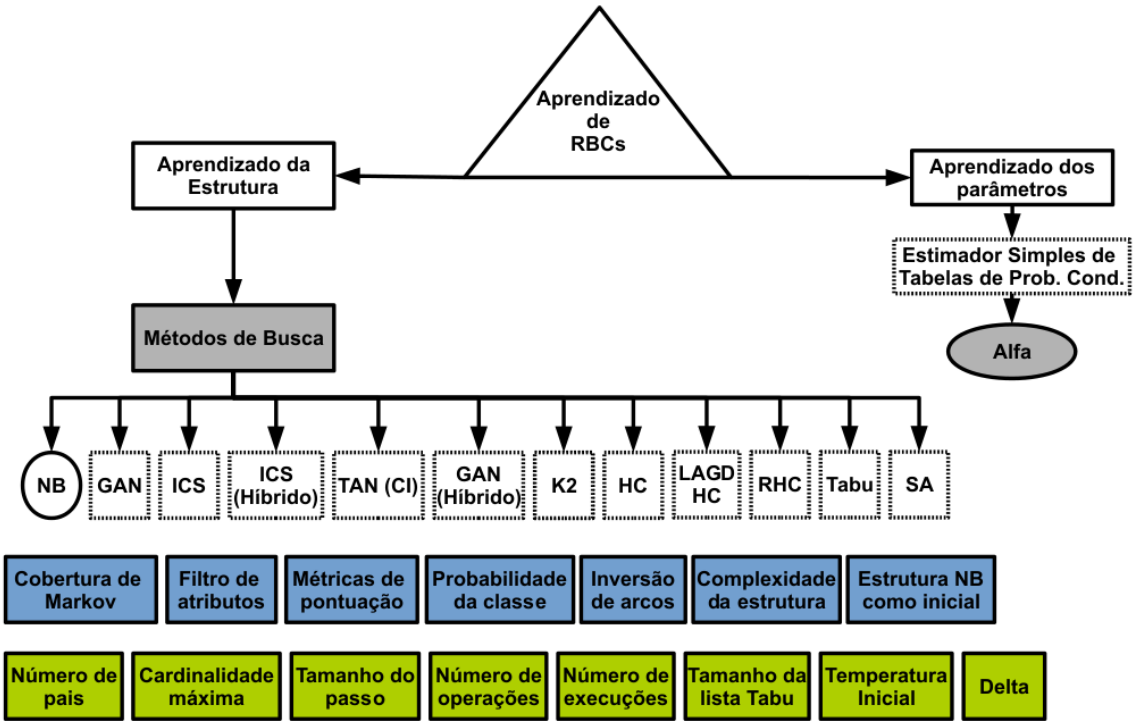


Figura 3.2: Processo de aprendizado em RBCs: foco no aprendizado da estrutura.

Como visto nessa figura, o Naïve Bayes (NB) é o único método de busca representado por uma elipse, bem como o alfa do estimador de tabelas de probabilidades. A razão para isso é que os mesmos chegaram no nível mais baixo da árvore que abstrai os passos principais de aprendizado de RBCs. Portanto, afirma-se que o Naïve Bayes não possuirá outros componentes associados em seu método de criação de estrutura.

Além disso, na Figura 3.2 foram especificados dois níveis de componentes associados aos métodos de busca. O primeiro nível, ilustrado na figura pela cor azul, representa componentes executáveis pelo método de busca. Já o segundo nível, identi-

ficado na figura pela cor verde, define componentes que nada mais são que parâmetros dos métodos de busca de estrutura de RBCs. São componentes fortemente associados a alguns métodos e modificam principalmente a profundidade da busca a ser realizada.

Contudo, os métodos de busca não possuem todos os componentes associados aos dois níveis apresentados pela Figura 3.2. Por essa razão, a Tabela 3.1 determina de quais componentes os métodos de buscam dependem.

O aprendizado dos parâmetros RBCs, diferentemente do aprendizado da estrutura, possui apenas o parâmetro alfa (α) do estimador simples de tabelas de probabilidades condicionais como componente, o que consequentemente determina que todos os algoritmos de RBC gerados possuam o mesmo como componente associado. Por essa razão, α do estimador foi separado e destacado na tabela.

Tabela 3.1: Relação de dependência entre método de busca, componentes associados e alfa do estimador.

Componentes Associados	**Componente – Método de Busca**											
	NB	**TAN (CI)**	**ICS**	**ICS (H)**	**GAN**	**GAN (H)**	**BG K2**	**HC**	**LAGD HC**	**RHC**	**Tabu**	**SA**
MBC		X	X	X	X	X	X	X	X	X	X	X
Filtro		X		X	X	X	X	X	X	X	X	X
Métricas				X	X	X	X	X	X	X	X	X
Prob. Classe					X	X	X	X		X	X	X
Complexidade					X	X						
Cardinalidade			X	X								
Inversão de Arcos							X	X	X	X	X	
Nro de pais							X	X	X	X	X	
Estrutrura NB							X	X	X	X	X	
Tam Passo									X			
Nro de Operações									X			
Execuções										X	X	X
Lista Tabu											X	
Temperatura Inicial												X
Delta												X
Componente – Aprendizado dos Parâmetros												
Alfa do Estimador	X	X	X	X	X	X	X	X	X	X	X	X

Dada essas definições gerais do aprendizado de RBCs, nos próximos itens os componentes serão sucintamente descritos para que se possa ter uma visão geral de seu espaço de busca:

(A) **Método de busca:** Determina o procedimento que constrói a estrutura da RBC e pode representar qualquer método de busca tradicional, como uma busca gulosa, *Simulated Annealing* e outros. Entretanto, as funcionalidades dos métodos de busca dependem fortemente do tipo de relação entre os atributos que é almejada. As principais diferenças entre esses métodos para distinguir as estruturas estão no modo em que os arcos (entre dois nós) são adicionados, removidos e invertidos, se

os métodos são locais ou globais e se são baseados em restrições, em métricas de pontuação ou híbridos.

Mesmo que hajam essas diferenças, a estrutura final criada também é um fator de distinção. Existem algoritmos que não criam relações entre os atributos previsores, outros chegam no máximo em uma relação na forma de uma árvore (ou uma floresta) para representar a cadeia de influências entre os atributos, e os demais são flexíveis para poder criar qualquer tipo de grafo não-acíclico direcionado com a mesma finalidade.

Os itens abaixo descrevem sucintamente os componentes apresentados na Figura 3.2 e quais características estão presente na sua estrutura funcional. É importante frisar novamente que todos os métodos necessitam do α do estimador (item Q) para aprender a RBC.

- **Naïve Bayes (NB):** É um tipo de busca onde a estrutura será sempre fixa, com a criação de arcos partindo do atributo classe para os atributos previsores. A mudança de estrutura ocorre quando a quantidade de atributos é variada.
- **Tree Augmented Naïve Bayes (TAN) by Conditional Independence (CI) tests:** Versão do algoritmo Tree Augmented Naïve Bayes baseado em restrições, onde as arestas da árvore são geradas pela execução de testes de independência condicional. Possui dependência para os seguintes componentes: MBC (item B) e Filtro de Atributos (item C).
- ***Inductive Causation Search* (ICS):** Realiza operações baseando-se no algoritmo ICS, o qual realiza dois passos principais: primeiro encontra um esqueleto da rede, que é um grafo não-direcionado com arestas presentes e, em segundo, direciona todas as arestas no esqueleto para gerar o grafo da RBC. Este método de aprendizado de RBCs usa testes de independência para encontrar um esqueleto. Nesse tipo de algoritmo é utilizada uma variação do teste de independência condicional, onde métricas de pontuação são utilizadas para criar uma heurística que define o resultado do teste. Depende dos componentes MBC (item B), Cardinalidade (item F) e Métricas de Pontuação (item D − Métricas Locais).
- **ICS (H):** Um modificação do ICS original que também utiliza métricas de pontuação para definir quais nós são dependentes da variável classe, tornando-o uma versão híbrida do ICS original. Usa métricas de pontuação para determinar as relações dos atributos previsores em relação à variável

classe e a utiliza a mesma heurística do teste de independência condicional para formar a estrutura da RBC. Esse método é similar a sua versão original, contudo também necessita do Filtro de Atributos (item C) para determinar as dependências para a variável classe.

- ***General Augmented Naïve Bayes* (GAN):** É um método genérico para criação de uma estrutura da RBC em que as relações entre os atributos previsores são definidas por uma estrutura que pode ter a complexidade máxima de uma árvore ou uma floresta. Cria a RBC somente considerando métricas de pontuação locais e globais, tanto para criar as arestas da árvore, quanto para definir quais nós são dependentes da classe. Têm os seguintes parâmetros: MBC (item B), Filtro de Atributos (item C), Métricas de Pontuação (item D − Locais ou Globais), probabilidade da classe da RBC (item E − no caso das métricas globais) e a Complexidade (item G)

- **GAN (H):** Versão híbrida do método de busca GAN e o modifica por simplesmente criar as arestas da árvore utilizando-se de um método que aplica um teste de independência condicional puro (sem utilizar métricas de pontuação como estimativa do teste). Os componentes associados a esse método são os mesmos da sua versão original.

- ***Hill Climbing* (HC):** Utiliza uma abordagem gulosa para adicionar, remover e inverter arestas direcionadas no grafo da RBC. Também considera as arestas da estrutura do Naïve Bayes como parte da estrutura a ser removida, ou seja, remove as arestas de ligação de atributos previsores e a classe. Esse método possui outros seguintes parâmetros associados: MBC (item B), Filtro de Atributos (item C), Métricas de Pontuação (item D − Locais ou Globais), probabilidade da classe da RBC (item E − no caso das métricas globais), Inversão de Arcos (item H), Estrutura do Naïve Bayes como Inicial (item I) e Número Máximo de Pais (item J).

- ***Look Ahead in Good Directions Hill Climbing* (LAGD HC):** É uma abordagem que verifica uma sequência de estruturas futuras, ponderando verificar qual caminho será o melhor a ser seguido pelo algoritmo. Além dos componentes do *Hill Climbing*, possui um Tamanho do Passo (item K) e um Número de Operações (item L).

- ***Repeated Hill Climbing (R HC)*:** Ao invés de simplesmente aplicar um *Hill Climbing*, perturba aleatoriamente a estrutura e repete o processo para um número de iterações (repetições) gerenciado. Além dos parâmetros do

Hill Climbing, um Número de Execuções (item M) também é definido para esse método.

- **BG K2:** Aprende a RBC dos dados a partir de uma busca gulosa específica ao algoritmo K2. Cria estruturas mais gerais e necessita também de métricas de pontuação para guiar a busca. A sua diferença para o *Hill Climbing* é que a BG K2 considera apenas adição de arestas durante sua busca, não removendo-as do grafo que define a RBC naquele momento. Contudo, os seus parâmetros são os mesmos do método HC.
- **Busca Tabu:** É um algoritmo de otimização de propósito geral. A busca cria inicialmente uma solução arbitrária e recursivamente tenta selecionar uma nova solução vizinha melhor (utilizando métricas de pontuação) que a solução atual. A Busca Tabu possui adicionalmente um Número de Execuções (item M) e uma Lista Tabu (item N) em relação à HC.
- ***Simulated Annealing* (SA):** É um algoritmo de otimização combinatória que também é de propósito geral. Consiste em uma técnica local probabilística e se fundamente em uma analogia do processo de tempera de metais. Como componentes relacionados à busca SA estão: MBC (item B), Filtro de Atributos (item C), Métricas de Pontuação (item D − Locais ou Globais), probabilidade da classe da RBC (item E − no caso das métricas globais), Número de Execuções (item M), Temperatura Inicial (item O) e δ (item P).

(B) **Markov Blanket Classifier (MBC):** É um componente do tipo booleano, que quando assume valor verdadeiro e a estrutura da RBC é aprendida, uma correção por cobertura de Markov é aplicada dentro da estrutura da rede. Isso assegura que todos os nós na rede são parte da cobertura de Markov do nó de classificação (classe). Para maiores detalhes, veja a Subseção 2.1.1

(C) **Seleção e Descarte (Filtro) de Atributos:** É um componente associado ao trabalho de Sacha [1999b]. Define uma filtragem de atributos inicial que delimita quais nós são dependentes da variável classe. Essa filtragem também define quais atributos serão considerados na busca, pois a filtragem pode tanto retirar o nó e a aresta que o liga à classe, como também pode selecionar e remover apenas a aresta, deixando-o participar da busca. Possui três valores possíveis: vazio, seleção e descarte. O valor vazio não releva nenhum operador, fazendo com que todos os atributos sejam dependentes da classe e participem da busca. A seleção apenas remove as dependências entre atributos previsores em relação à classe, caso essa remoção aumente a métrica de pontuação da rede ou do vértice. Contudo, somente

as arestas são removidas e os atributos que eram dependentes da classe permanecem, participando do do restante da busca. Já o descarte, filtra os atributos previsores que são dependente classe, de forma causal, se os mesmos diminuírem o valor da métrica de pontuação associada e, se algum previsor for filtrado, o mesmo não vai participar do restante da busca.

(D) **Métricas**: São métricas utilizadas para guiar os métodos de busca durante sua sequência de passos para encontrar uma RBC. Possuem dois tipos, locais e globais, que são devidamente descritas nos trabalhos de Bouckaert [1995], Bouckaert et al. [2013], Witten et al. [2011] e Sacha [1999b]. Os próximos itens especificam cada métrica:

- **Métricas de pontuação locais:** Aprender a estrutura de um RBC pode ser considerada um problema de otimização, onde a métrica de pontuação Q da estrutura da rede, dado um conjunto de dados D, $(Q(RB_D|D))$ precisa ser maximizada. A métrica de pontuação pode ser baseadas em várias técnicas presentes na literatura. As que foram utilizadas aqui são: Bayesiana, Entropia, BDeu (*Bayesian Dirichlet equivalence uniform*), MDL (*Minimum Description Length*) e AIC (*Akaike Information Criterion*). A localidade dessas métricas é referente a como a avaliação é feita. Métricas locais têm uma propriedade prática em que a pontuação de toda rede pode ser decomposta como a soma (ou o produto) de todos as pontuações dos nós individuais.
- **Métricas de pontuação globais:** Para métricas globais, a pontuação da rede é calculada sobre toda a estrutura da RBC. No total, foram utilizadas nove (9) métricas de pontuação, e com variações nos parâmetros das próprias métricas chegou-se a um total de 17 variações de métricas globais de pontuação, que são: LOO-CV (*Leave One Out Cross-Validation*), k-Fold-CV (*k-Fold Cross-Validation* com valor de k igual a 5 e 10), Cumulative-CV (*Cumulative Cross-Validation*), HGC (*Heckerman-Geiger-Chickering* com e sem o uso de Prior), SB (*Standard Bayesian*), LC (*Local Criterion*), LogC (Logaritmo da Probabilidades das Classes), LLOO (Local Leave One Out Cross-Validation) e LKCV (*Local k-Fold Cross-Validation* com variação na quantidade de partições em 5 e 10, e na quantidade de execuções em 1, 5 e 10 vezes).

(E) **Probabilidade em relação à variável classe (Prob. Classe):** A probabilidade está definida apenas para métricas de pontuação globais e possui tipo

booleano. A probabilidade em relação a variável classe é determinada para métricas de pontuação globais porque as mesmas precisam, a cada passo do algoritmo de busca, avaliar a estrutura da RBC. Isso é feito analisando-se o modelo globalmente. Assim, caso for configurada como falso, a acurácia do classificador que define a métrica global de pontuação somente é aumentada se o classificador retornar exatamente a classe correta (*zero-one loss*). Caso contrário, a probabilidade em relação à classe é retornada para estimar a acurácia e, consequentemente, o valor da métrica de pontuação.

(F) **Cardinalidade Máxima:** Algoritmos baseados em restrições utilizam em sua busca testes de independência condicional para criar a estrutura. Eles requerem um parâmetro que é a cardinalidade máxima. A ideia dessa classe de algoritmos é testar se cada par de variáveis (ou atributos) da base de dados, como x e y, são condicionalmente independentes dado um conjunto de variáveis Z. Z é definido como o subconjunto de nós que são vizinhos de ambas variáveis x e y no grafo causal da RBC. Se o método de busca usado para construir a rede identifica uma independência $(x, y|Z)$, a aresta entre x e y não é criada ou é removida. O valor máximo para cardinalidade determina a largura limite do subconjunto Z a ser considerado nos testes de independência condicional $(x, y|Z)$. O valor inteiro inicial e limite desse componente são respectivamente iguais a 1 e 10.

(G) **Complexidade máxima da estrutura:** Também é um componente booleano e se tiver valor verdadeiro permite uma maior flexibilidade da busca, permitindo que a RBC gerada possa ser mais complexa nas relações entre atributos. Quando esse componente possui valor verdadeiro, o método de busca pode criar uma estrutura que é no máximo uma floresta. Caso contrário, pode chegar a uma árvore de relações. Isso não quer dizer que os métodos não possam criar uma estrutura como o Naïve Bayes ou algo mais simples que uma árvore ou uma floresta.

(H) **Inversão dos Arcos:** É um componente booleano, que quando assume valor verdadeiro os arcos invertidos também são considerados no próximo passo do método de busca.

(I) **Estrutura inicial como Naïve Bayes (Estrutura NB):** É um componente booleano que, quando verdadeiro, a estrutura inicial da rede utilizada para aprender a estrutura completa é um Naïve Bayes. Quando configurado como falso, uma rede vazia é declarada como estrutura inicial. É importante ressaltar que esse componente, quando declarado como verdadeiro, é utilizado somente como uma

estrutura inicial para certos métodos e os mesmos podem modificar bruscamente essa estrutura a fim de gerar um modelo mais condizente aos dados.

(J) **Quantidade máxima de nós definidos como pais (Nro de Pais):** Para o tipo de algoritmo baseado em pontuação, restringe-se o número de pais máximo que um nó da RBC pode ter. Os pais de um nó influenciam de forma causal o nó que está sendo verificado. Esse componente reflete diretamente no tipo de estrutura a ser criada, pois restringe a busca realizada para construir a estrutura da RBC. O número de pais está definido no intervalo inteiro $[1; 10]$.

(K) **Tamanho (Tam) do Passo:** Componente utilizado somente no método LAGD *Hill Climbing*. O tamanho do passo significa ter o controle sobre a profundidade da busca. Por exemplo, caso o tamanho do passo possua valor igual a dois (2), significa que todas as estruturas em uma distância de dois (da estrutura atual da RBC) são levadas em consideração para a decisão de quais arcos adicionar, remover ou inverter. Possuindo valor igual a um (1) resulta em um *Greedy Hill Climbing*. É um componente do tipo inteiro que varia entre um (1) e cinco (5).

(L) **Número (Nro) de operações:** Também presente apenas no LAGD *Hill Climbing* e é um componente que configura o número de operações por tamanho do passo. Por exemplo, se assumir o valor inteiro cinco (5), significa que para os próximos passos, que se tem visão futura, somente as cinco melhores operações (adição, remoção e inversão de arestas) são levadas em consideração para o cálculo da melhor sequência de operações para criação da estrutura da RBC. Possui valor definido no intervalo inteiro $[1; 10]$.

(M) **Número de Execuções:** Esse componente possui vários significados e valores que podem ser assumidos, dependendo fortemente do método de busca relacionado. Para o método *Repeated Hill Climbing* configura o número de vezes que o *Hill Climbing* é executado para retornar a melhor estrutura dentro dessas execuções. Possui um valor inteiro no intervalo $[1; 10]$ considerando apenas 10 valores possíveis nesse intervalo.

Já para a Busca Tabu significa o número de passos a ser executado (ou iterações) para percorrer o espaço de busca de estruturas. Tem tipo e intervalo iguais ao *Repeated Hill Climbing*.

Por fim, para o *Simulated Annealing* também define o número de iterações a ser executada pela busca. Contudo, o intervalo é maior $[10; 250]$ podendo atingir 25 valores.

(N) **Tamanho da Lista Tabu:** É um componente singular do método Busca Tabu. O tamanho da lista Tabu determina quantas iterações podem ser feitas para retornar para um ótimo local. Quanto maior tamanho dessa lista, maior será o tempo será consumido para retornar para o ótimo local, que terá maior probabilidade que não será visitado novamente. A solução retornada pela busca Tabu é a melhor solução durante a execução do algoritmo. Está definida no intervalo $[1; 10]$.

(O) **Temperatura Inicial:** Somente utilizado pelo método *Simulated Annealing*, é a temperatura inicial do sistema. A analogia a temperatura do metal é utilizado por essa busca local para tentar escapar de ótimos locais. Quando se tem temperatura alta, permite-se com maior probabilidade que se aceite soluções que degradam a função objetivo. Com o passar do tempo, a temperatura vai diminuindo e o movimentos bruscos de soluções vão se tornando menos aceitáveis (probabilisticamente). O intervalo real $[10,0; 300,0]$ define a temperatura inicial, onde existem no máximo 10 valores a serem assumidos pela mesma.

(P) **Delta (δ):** É o fator utilizado para atualizar a temperatura T do *Simulated Annealing*, considerando que $T_{i+1} = T_i \cdot \delta$. Pode assumir valores no intervalo real $[0; 1]$, com 10 valores possíveis.

(Q) **α do Estimador:** Tendo-se a estrutura final da RBC, o estimador é utilizado para montar a tabela de probabilidades condicionais de todos os nós da rede Bayesiana e, assim, estimar as probabilidades condicionais da variável classe pertencer a determinadas classes. Um estimador simples é utilizado e o mesmo produz estimativas diretas dos dados a partir de probabilidades condicionais, limitada pela estrutura da rede. A Equação 3.1 determina como é feito o cálculo por esse estimador:

$$P(x_i = k|pa(x_i) = j) = \frac{N_{ijk} + N'_{ijk}}{N_{ij} + N'_{ij}} \tag{3.1}$$

Na Equação 3.1, x_i é um atributo previsor discreto com valor igual a k e $pa(x_i)$ são os precedentes causais ("pais") de x_i assumindo um valor j. As definições das variáveis do lado direito da equação são descritos nos itens abaixo:

- N_{ijk}: é definido como o número de casos na base de dados D onde a variável aleatória X_i está na configuração k e seus "pais" $Pa(x_i)$ estão na configuração j.

- N_{ij}: é o número de casos em D onde os pais da variável aleatória X_i está na configuração j. Denota-se que: $N_{ij} = \sum_{k=1}^{r_i}(N_{ijk})$, onde r_i é o número de configurações da variável X_i.
- N'_{ijk}: é o parâmetro α do estimador. N'_{ijk} e N'_{ij} representam escolhas dos Priors na contagens, que são restringidas por: $N'_{ij} = \sum_{k=1}^{r_i}(N'_{ijk})$. Ou seja, o α é um parâmetro do estimador que ode ser interpretado como a contagem inicial de cada valor de probabilidade. No AG, seu valor varia entre em 0,000001 e 9,000, como determinado pelo trabalho de Sacha [1999b]. Dentro desse intervalo, existem 1.000 (mil) valores reais possíveis que o α pode assumir. O valor 0,000001 é utilizado pois os algoritmos do *framework* utilizado não permitem que o valor desse componente seja estritamente igual a zero.

As tabelas de probabilidade condicionais são construídas para cada atributo da base de dados seguindo a Equação 3.1 e a quantidade de valores dessa tabela depende da estrutura previamente criada por um método de busca. Com as tabelas de probabilidade em cada nó da rede, é possível estimar qual classe é mais provável dados os valores especificados (nos exemplos) para os atributos previsores.

É importante ressaltar que pretende-se futuramente utilizar-se de outros estimadores de RBs adaptados para RBCs. Essa tarefa não foi feita até o presente momento devido às dificuldades de projeto de tais estimadores para RBCs. Assim, versões futuras do sistema poderão contar com mais opções para esse componente.

3.3 Representação do Indivíduo

Após o estudo e identificação dos principais componentes vistos na última seção e no capítulo anterior (Seção 2.2), pode-se associá-los à representação do indivíduo do AG.

Com o diagrama da Figura 3.2 e as dependências definidas pela Tabela 3.1, pode-se afirmar, nesse momento, que a representação do indivíduo utilizada para o AG é dinâmica e está de acordo com as dependências entre componentes. Em outras palavras, a representação é modificada de acordo com o algoritmo que gera a estrutura da RBC.

No entanto, o indivíduo é sempre representado por um vetor real com 11 posições, com valores no intervalo real $[0, 1]$. Cada posição do cromossomo do indivíduo abstrai um componente, que em alguns casos pode ser ignorado durante o mapeamento. Representações de algoritmos distintos podem considerar componentes também distintos, deixando certas posições de sua representação real não-funcionais.

A Figura 3.3 define o que pode representar cada posição do indivíduo, ou seja, os seus possíveis fenótipos e quais posição são não-funcionais. As posições não-funcionais

são destacadas na figura pela cor cinza e pela ausência da definição do seu componente associado.

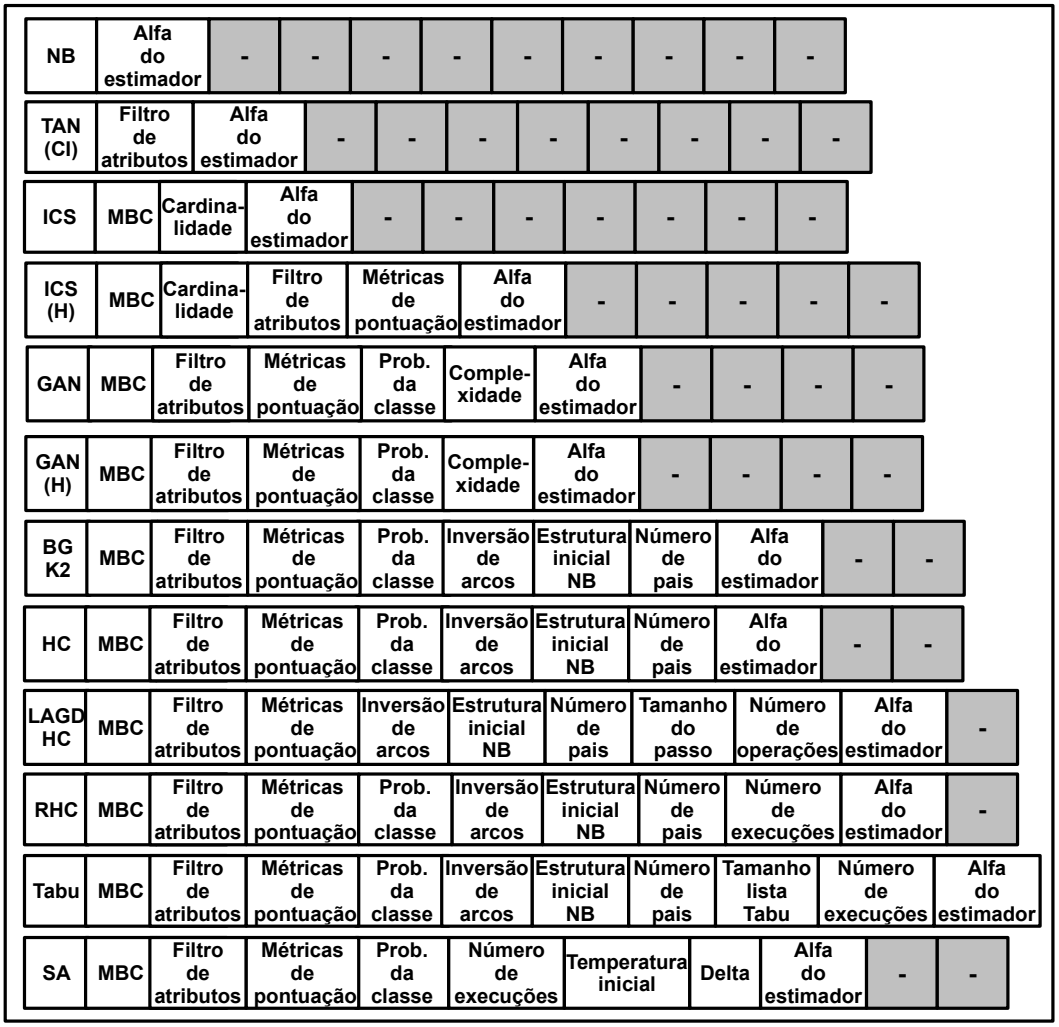


Figura 3.3: Possíveis fenótipos dos indivíduos dados os componentes identificados.

É válido mencionar que as únicas posições funcionais que sempre estarão presentes no vetor do indivíduo são o método de busca e o alfa do estimador. Por exemplo, quando o Naïve Bayes é mapeado como o componente de método de busca, o mesmo não possuirá demais componentes associados, alocando somente um valor de α para estimar as tabelas de probabilidades condicionais.

Um caso oposto é quando a Busca Tabu é mapeada como componente do método de busca de estrutura de RBCs. Esse método considera todas as 11 posições do vetor, o qual representa o genótipo do indivíduo, como funcionais. Na Tabela 3.1 é visto que a Busca Tabu possui um maior número de componentes associados o que consequentemente gera uma alocação completa do vetor que representa o indivíduo.

Já quando os demais métodos métodos de busca são mapeados, os mesmo ficam em um meio termo entre Naïve Bayes e Busca Tabu por posições funcionais, como também é visto na Figura 3.3.

3.4 Função de Avaliação

A fim de avaliar o quão efetivo são os algoritmos de RBC gerados pelo AG, os classificadores representados por cada indivíduo são criados e executados sobre um conjunto de bases de dados (ou sobre uma base de dados específica) para gerar um modelo de RBC para esses dados. A Figura 3.4 mostra o processo completo de avaliação de um dado indivíduo.

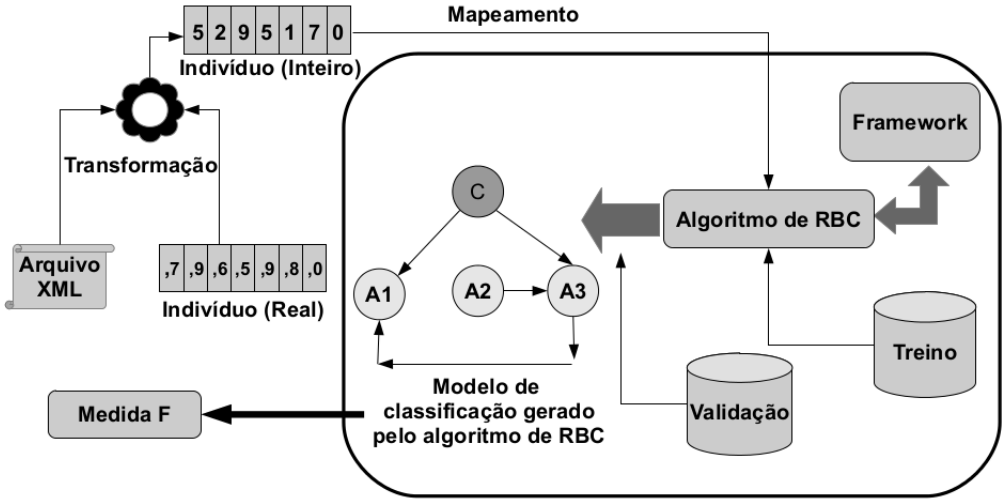


Figura 3.4: Processo de avaliação de um indivíduo.

Todos as etapas do processo de avaliação dos indivíduos, ilustrados na Figura 3.4, serão detalhadas nos próximos parágrafos.

Inicialmente, cada indivíduo possui uma representação real, onde cada posição do cromossomo do indivíduo está no intervalo $[0; 1]$ e determina o uso de uma opção de um dado componente. Conjuntamente com o arquivo *XML* de definição das dependências entre componentes (abstraído pelas Figura 3.2 e Tabela 3.1), o cromossomo real é transformado em um cromossomo inteiro de mesmo tamanho.

Em geral, o número real é multiplicado pela quantidade máxima de opções (valores possíveis que pode assumir) daquele componente, resultando em um número inteiro arredondado. Por exemplo, dentro dos componentes avaliados anteriormente, existem cinco métricas de pontuação local, que são: {Bayesiana (1º), MDL (2º), BDeu (3º),

AIC (4º), Entropia (5º)}. Considere que a ordem das métricas dentro do conjunto seja relevante para a tomada de decisão.

Quando o componente de métrica de pontuação local precisar ser mapeado, será retornado o valor real ($[0; 1]$) dentro de sua posição associada ao cromossomo. Esse valor real é multiplicado pelo tamanho do conjunto. Suponha que o número real seja 0.2 e o tamanho do conjunto de métricas seja igual a cinco (5), como já mencionado. O valor $0,2 \times 5 \approx 1$, o qual mapeia a primeira posição do conjunto: a métrica Bayesiana. Se o número encontrado fosse $0,65$, significaria que a terceira posição (BDeu) seria mapeada, pois $0,65 \times 5 \approx 3$.

A única exceção é para o componente método de busca, em que uma faixa de valores justa é criada para cada método. Para que os métodos de busca fossem mapeados de forma justa, uma contagem de seus parâmetros foi realizada. A motivação para o uso dessa contagem é para destacar a diferença entre as probabilidades dos métodos. Isto é, métodos de busca com quantidade de parâmetros diferentes possuem probabilidades distintas de serem escolhidos pelo AG para participar do cromossomo do indivíduo.

O valor real presente na posição desse componente é convertido para um valor inteiro da seguinte forma: um intervalo inteiro de valores é criado para cada método de busca, iniciada em zero, e de tamanho igual à quantidade de parâmetros avaliados. Como cada método possui um intervalo, o valor real é multiplicado pela soma total de todos os valores de componentes dos métodos. Assim, o valor gerado é comparado com os intervalos e o método de busca escolhido é aquele em que o valor pertencer ao seu intervalo. A Tabela 3.2 estabelece as quantidades de componentes para cada método de busca utilizado no presente trabalho. É importante mencionar que na Tabela 3.2 os métodos foram separados de acordo com as suas características híbridas e de acordo com suas métricas de pontuação local e global.

Por exemplo, para o Naïve Bayes a quantidade de componentes é igual a 1.000 porque o α do estimador é o único parâmetro de que esse método de busca depende e o mesmo possui 1.000 valores possíveis, como mencionado anteriormente. Para os demais casos, outros parâmetros foram levados em consideração, causando diferenças entre as contagens.

Nesta forma de representação dinâmica do indivíduo, é provável que um dado gene (posição do cromossomo) não se manifeste, dependendo da quantidade de parâmetros associados ao componente método de busca. Caso isso ocorra, o valor -1 é mapeado para aquele gene.

Para definir o algoritmo de RBC de acordo com o indivíduo, dois *frameworks* foram utilizados neste trabalho para prestar suporte durante o mapeamento do indivíduo

Tabela 3.2: Métodos de Busca e suas quantidades de componentes.

Método de Busca	**Quantidade de Componentes**
Naïve Bayes	1.000
TAN (CI)	2.000
ICS	10.000
ICS (Híbrido)	30.000
GAN (Métricas locais)	60.000
GAN (Métricas globais)	408.000
GAN (Híbrido – Métricas locais)	60.000
GAN (Híbrido – Métricas globais)	408.000
Hill Climbing (Métricas locais)	120.000
Hill Climbing (Métricas globais)	816.000
LAGD Hill Climing (Métricas locais)	120.000
Repeated Hill Climbing (Métricas locais)	12.000
Repeated Hill Climbing (Métricas globais)	81.600
BG K2 (Métricas locais)	120.000
BG K2 (Métricas globais)	816.000
Busca Tabu (Métricas locais)	120.000
Busca Tabu (Métricas globais)	816.000
Simulated Annealing (Métricas locais)	3.000
Simulated Annealing (Métricas globais)	20.400
Total das contagens – Espaço de busca	4.960.000

e a posterior execução do algoritmo de RBC.

O primeiro deles é o jBNC (*Java Bayesian Network Classifiers*), desenvolvido por Sacha [1999a,b], que implementa diversos classificadores Bayesianos e já demonstrou estar apto a ser aplicado em uma variedade de bases de inteligência artificial, aprendizado de máquina e mineração de dados. O segundo *framework* utilizado é o WEKA (*Waikato Environment for Knowledge Analysis*), descrito em Hall et al. [2009] e reconhecido por ter uma diversidade de algoritmos de aprendizado de máquina.

Ressalta-se que as funcionalidades e algoritmos do jBNC foram incluídos (re-implementadas) no WEKA com o objetivo de complementá-lo e utilizar-se de apenas um *framework* mais robusto e flexível quanto a novos problemas de aprendizado de máquina.

Dado o indivíduo com valores inteiros em seu cromossomo, o mesmo é mapeado para um algoritmo de RBC, onde cada número inteiro em uma posição do indivíduo é a opção ou o valor do componente representado em ordem por uma lista de valores presente no *XML*. Para lidar com o algoritmo de RBC criado, uma modificação no *fra-*

mework WEKA foi realizada, e esse passa a prestar suporte à execução dos algoritmos de RBCs dentro do AG. Portanto, os algoritmos construídos são executados em um conjunto de bases de dados de treino para induzir um modelo de RBC, o qual é então avaliado usando um conjunto de bases de dados de validação.

A função de avaliação para um indivíduo qualquer da população do AG é gerada por meio do conjunto de validação, usando a média da medida F (*F-measure*) [Witten et al., 2011] para aquele conjunto. A medida F é definida de forma matemática pela Equação 3.2:

$$medidaF = \frac{2 \cdot (\text{Precisão} \cdot \text{Revocação})}{(\text{Precisão} + \text{Revocação})} \tag{3.2}$$

A medida F é uma média harmônica entre precisão e revocação e é uma métrica interessante porque pondera diferentes níveis de desbalanceamento de classes e considera a precisão (o número de exemplos corretamente classificados sobre o número total de exemplos na base de dados analisada) e revocação (o número de exemplos corretamente classificados na classe c sobre o total de exemplos classificados como c, desprezando sua classe verdadeira). Com a finalidade de prevenir *overfitting*, o conjunto de treinamento é reamostrado a cada n gerações.

3.5 Cruzamento e Mutação

Mesmo com a representação do indivíduo sendo dinâmica, os operadores de cruzamento uniforme e mutação de um ponto podem ser facilmente aplicados. Isso porque esses operadores são direcionados ao genótipo de um indivíduo (representação no intervalo real [0;1]) e não sobre o fenótipo do indivíduo (representação dos indivíduos por componentes da Figura 3.3).

Portanto, primeiro os operadores são aplicados na representação real do indivíduo e, em seguida, os indivíduos são mapeados e avaliados como discutido na seção anterior.

A Figura 3.5 ilustra o processo de cruzamento uniforme aplicado em dois indivíduos quaisquer. A máscara de zeros (0's) e uns (1's) ao meio dos dois indivíduos é gerada de acordo com uma função de probabilidade uniforme. Caso a função retorne valores menores ou iguais a 0,5, o valor um (1) é assumido por aquela posição da máscara. Caso contrário, o valor zero (0) é alocado para aquela posição.

Assim, é possível combinar dois indivíduos de acordo com a máscara. O valor um (1) significa que ocorrerão trocas entre materiais genéticos e o valor zero (0) a manutenção dos mesmos.

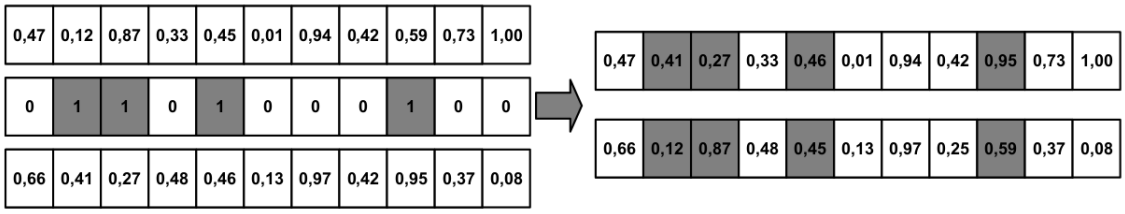


Figura 3.5: Cruzamento uniforme aplicado em dois indivíduos com representação real.

Já a Figura 3.6 exemplifica a utilização da mutação de um ponto em um dado indivíduo. Primeiramente, uma dentre as 11 posições possíveis é escolhida para a aplicação da mutação. Cada posição possui a mesma chance de ser escolhida e essa é a razão do nome desse tipo de mutação. Após a escolha, o valor da posição é alterado de forma aleatória dentro do intervalo plausível de cada posição, o qual está definido entre zero (0) e um (1).

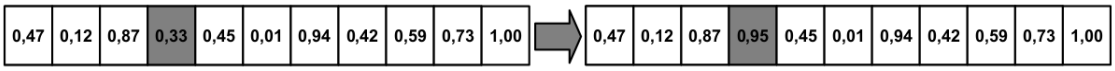


Figura 3.6: Mutação de um ponto aplicada em um indivíduo com representação real.

Nas Figuras 3.5 e 3.6, as posições do indivíduo destacadas em cinza são aquelas que sofreram trocas de material genético.

3.6 Adaptação do AG para lidar com Métricas de Complexidade

Esta seção apresenta uma modificação realizada no algoritmo genético para que o mesmo possa considerar as características de conjuntos de dados quando for gerar um algoritmo. Nessa adaptação, o AG será responsável por gerar algoritmos de RBC para grupos de bases de dados. Em outras palavras, almeja-se treinar o AG com um grupo de bases de treino para que ele aprenda as singularidades dos dados e, ao final, caso ele utilize as métricas de complexidade, o mesmo possa decidir qual algoritmo de RBC escolher para cada base no grupo de bases de teste. A adaptação será denominada AG-M, pois distingue da versão anterior e o faz incorporar as métricas também em seu nome.

O AG-M será guiado por métricas de complexidade, e seu funcionamento é similar ao uso de uma gramática em suas tomadas de decisões. Isto é, os componentes dos algoritmos são escolhidos caso as métricas de complexidade das bases de dados assumam valores específicos dentro de um intervalo. Por exemplo: *Se Métrica1 < 0,5*

então: Escolha o método de busca Naïve Bayes; *Caso contrário: Escolha o método de busca gulosa do K2.*

O foco nas métricas de complexidade de dados é porque elas são aptas a estimar a dificuldade dos problemas de classificação. Por meio de medidas de sobreposição de classe, geometria e separabilidade, é possível determinar o quão difícil é aprender aquele problema.

A biblioteca de complexidade de dados (DCoL) [Orriols-Puig et al., 2010] é um *framework* em C++ que provê a implementação de um conjunto de métricas projetadas para caracterizar a complexidade aparente de bases de dados de aprendizado supervisionado. A DCoL foi originalmente proposta por Ho & Basu [2002] e o trabalho foi continuado em Ho et al. [2006]. No total, a biblioteca possui 14 métricas distintas de complexidade, divididas em três vertentes, e descritas abaixo:

- **Métricas de sobreposição nos valores de atributos previsores de diferentes classes:** Essas métricas focam na capacidade dos atributos previsores separar os exemplos de diferentes classes.

    1. **A taxa discriminativa máxima de Fisher (F1):** Essa medida computa o poder discriminativo máximo de cada atributo. Um valor alto do discriminante de Fisher indica que, no mínimo, um dos atributos permitem aprender a separar os exemplos de diferentes classes com partições que são paralelas a um eixo no espaço de atributos. Um valor baixo dessa medida não significa que as classes não são linearmente separáveis, mas que elas não podem ser discriminadas por um hiperplano paralelo a um dos eixos do espaço de atributos. Essa métrica é implementada para problemas multi-classe.
    2. **A taxa discriminativa máxima de Fisher do vetor orientado (F1v):** Essa medida complementa F1 por buscar um vetor orientado que separa os exemplos em duas classes. Assim, F1v é implementada apenas para problemas com duas classes. Um alto valor de F1v indica que existe um vetor que pode separar os exemplos pertencendo a diferentes classes depois que as instâncias são projetadas no espaço.
    3. ***The overlap of the per-class bounding boxes* (F2):** Computa a sobreposição de caldas de distribuições definidas por instâncias de cada classe. A especificação dessa medida para bases de dados com duas classes é a seguinte: para cada atributo, compute a taxa de largura do intervalo de sobreposição (ie., o intervalo que existem instâncias de ambas as classes) para a largura do intervalo completo. A métrica retorna o produto das taxas calculadas

para cada atributo. Para problemas multi-classe, computa-se F2 para cada par de classes, obtém-se o valor absoluto de todas elas e retora-se a soma de todos esses valores. Um valor baixo de F2 significa que os atributos podem discriminar os exemplos de diferentes classes.

4. **A eficiência máxima dos atributos individuais (F3):** Mensura o poder discriminativo individual de atributos previsores isolados e retorna o valor do atributo que pode discriminar o maior número de instâncias de treino. Aplicável para bases de dados em que a quantidade de classes é maior que dois (m $>$ 2).
5. **A eficiência conjunta dos atributos (F4):** Segue a mesma ideia apresentada por F3, mas leva em conta o poder discriminativo de todos os atributos (portanto, uma eficácia onjunta de previsores). A F4 pode prover mais informação por considerar todos os atributos desde que se quer exaltar o poder discriminativo todos eles conjuntamente. Aplicável para problemas onde as bases de dados possuem quantidade de classes m $>$ 2.

- **Métricas de separabilidade de classe:** Essas medidas estimam o quanto as classes são separáveis por examinar o comprimento e a linearidade de suas fronteiras.

  1. **A soma minimizada da distância do erro de um classificador linear (L1):** Essa métrica pondera até que ponto os dados de treinamento são separáveis. Para esse propósito, retorna a soma da diferença entre a predição de um classificador linear (*Support Vector Machines*) e o valor atual da classe. Essa medida é implementada apenas para problemas com duas classes. Um valor zero para L1 indica que o problema é linearmente separável.
  2. **O erro de treinamento de um classificador linear (L2):** Também provê a informação do quanto o conjunto de treinamento é linearmente separável. L2 constrói um classificador linear como explicado acima (L1) e retorna o erro de treino. Como antes, essa medida é apenas aplicada para bases de dados com duas classes.
  3. **A fração de pontos na fronteira de classes (N1):** Oferece uma estimativa do comprimento da fronteira de classe. Altos valores de N1 determinam que a maioria dos pontos estão estabelecidos na fronteira de classes e, portanto, pode ser uma tarefa difícil aprender um classificador que define a fronteira de classes de forma precisa.

4. ***The ratio of average intra/inter class nearest neighbor distance (N2):*** Compara a extensão (comprimento) intra-classe com as distâncias para vizinhos mais próximos de outras classes. Baixos valores de N2 sugerem que os exemplos da mesma classe estão próximos no espaço de atributos. Altos valores definem uma provável dispersão dos exemplos da mesma classe.

5. **A taxa de erro *leave-one-out* de um classificador Um Vizinho Mais Próximo (N3):** Denota o quão perto estão os exemplos de diferentes classes. Retorna a taxa de erro *leave-one-out* do 1-NN (o classificador K-Vizinhos Mais Próximos com K=1). Baixos valores de N3 indicam que há uma grande lacuna na fronteira das classes.

- **Métricas de Geometria, Topologia e Densidade de *Manifolds*:** Essas métricas apresentam uma caracterização indireta da separabilidade de classes. Elas assumem que o problema é composto por vários *manifolds* espalhados por cada classe. O formato, posição e conectividade desses *manifolds* oferecem algumas dicas de quão bem as classes são separadas na densidade ou população de cada *manifold*.

  1. **A não-linearidade de um classificador linear (L3):** . Dado a base de dados de treinamento, o método cria um conjuto de teste por interpolação linear com coeficientes aleatórios entre pares de instâncias da mesma classe selecionadas de forma arbitrária. Então, a medida retorna a taxa de erro do teste do classificador linear (*Support Vector Machine* com função de *kernel* linear) treinado com o conjunto de treinamento original. L3 é sensitiva a suavidade da fronteira classificadora e a sobreposição sobre envoltória convexa das classes. A medida L3 é implementada apenas para problemas com duas classes.

  2. **A não-linearidade de um classificador Um Vizinho Mais Próximo (N4):** Cria um conjunto de teste proposto por L2 e retorna o erro de um classificador 1-NN.

  3. **A fração de esferas da cobertura máxima (T1):** Essa medida foi descrita pela ideia das formas dos *manifolds* das classes com a noção de subconjunto de aderência. Um conjunto de aderência é uma esfera centrada em um exemplo do conjunto de dados que cresce o máximo possível até encostar em um exemplo de outra classe. Portanto, um conjunto de aderência contem um conjunto de exemplos da mesma classe e não pode crescer sem

incluir exemplos de outras classes. T1 considera somente o maior conjunto de aderência ou esferas, removendo todos os outros que estão incluídos nos demais. Então, essa métrica retorna o número de esferas normalizadas sobre o número total de pontos.

4. **O número médio de pontos por dimensão (T2):** Retorna a taxa do número de exemplos na bases de dados em relação ao número de atributos. É um indicador grosseiro de esparsidade da base de dados.

Dada as explicações das métricas de complexidade, relacionou-se algumas dessas métricas no impacto da definição de alguns componentes de RBC descritos na seção anterior. O que estamos almejando é uma relação entre o algoritmo de RBC gerado (junção de componentes) e as características dos dados (métricas). A relação entre componentes de RBCs e métricas de complexidade estão descritas nos próximos itens:

- **Métricas de sobreposição nos valores de atributos previsores de diferentes classes** − Podem ser úteis em pelo menos quatro situações, definidas abaixo:

  (1) Na escolha dos métodos de busca para criação da estrutura da RBC;

  (2) Na determinação da complexidade máxima da estrutura da RBC;

  (3) Na definição da cardinalidade máxima dos métodos de busca baseados em restrições;

  (4) Na quantidade máxima de nós definidos como precedentes causais (número máximo de pais).

  Em um primeiro experimento, considerou-se apenas F2 para a situação 1. Um baixo valor da mesma implica que os atributos podem independentemente discriminar os exemplos em diferentes classes. Ou seja, quando isso ocorrer, a estrutura de um pode necessitar de modelos mais simples, como não ocorrer relações entre os atributos previsores ou poucas relações causais.

  A métrica de complexidade discriminativa F3 foi associada a segunda situação. A justificativa para tal uso é similar à explicação do mapeamento entre componente de método de busca e F2. Quando se tem uma RBC em que as relações entre os atributos são ditas menos complexas (como aquelas criadas pelo TAN), a quantidade de arestas criadas por seu indutor é distinta da quantidade de arestas geradas por um indutor mais flexível, que gere arestas mais complexas (como o FAN ou BG K2), na maioria dos casos. O último indutor possui uma maior

flexibilidade na criação do conjunto de arestas e não é tão rígido ao definir a estrutura da rede. Isso implica que uma métrica discriminativa individual pode mapear bem esse fato, pois considera o poder discriminativo individual dos atributos, podendo ou não gerar mais arestas para a estrutura da RBC.

A métrica F4, a qual é baseada na eficácia conjunta de atributos, mapeou as demais situações. Ambas situações são caracterizadas por componentes (parâmetros) que limitam o espaço de busca do algoritmo, ou seja, o subconjunto de atributos que pode, respectivamente, d-separar outros atributos e a quantidade de atributos máxima que qualquer nó pode ter como predecessor causal (pai). A eficiência coletiva de atributos pode mapear bem os conjuntos de atributos que podem influenciar outros. Assim, quando se tem uma métrica F4 com valor alto/baixo, isso pode significar que o conjunto de atributos discrimina bem/mal as classes. A cardinalidade e a quantidade de pais lidam com subconjuntos de atributos que podem influenciar outros. Ao final do algoritmo, as ligações da RBC serão influenciadas pela cardinalidade e pela quantidade de pais, o que resultará na eficiência conjunta dos atributos da RBC.

- **Métricas de separabilidade de classes** – Podem ser utilizadas em pelo menos três casos, os quais são definidos nos itens a seguir:

  (1) Para a utilização de uma cobertura de Markov ao final do aprendizado da estrutura.

  (2) Para definição das métricas de pontuação locais e globais.

  (3) Na especificação do uso de seleção/descarte (filtro) de atributos.

  Para o primeiro caso, uma métrica de complexidade baseada no erro de classificação (N3) é utilizada. A justificativa para isso é que, quando se utiliza MBC como parâmetro do método de busca, uma diferença no erro de classificação é observada. Portanto, uma métrica baseada no nesse erro pode ser boa escolha de mapeamento.

  No segundo caso, uma métrica relacionada à fronteira de classe (N1) é definida para mapear esse componente. A ideia de utilizar a métrica N1 para o componente de métricas de pontuação está relacionado com a ideia desse componente ajudar iterativamente o método de busca a construir a RBC. Quando o valor da métrica N1 é muito alto, pode ser mais difícil aprender a fronteira de classe de forma precisa. Com isso, quando se tem uma métrica de pontuação adequada, pode-se melhorar o aprendizado.

Por fim, o terceiro caso está associado a uma métrica de complexidade que avalia a dispersão dos exemplos: N2. A ideia é associar a quantidade de conexões da RBC com a dificuldade de aprendizado do algoritmo em relação aos exemplos, definida pela métrica N2.

- **Métricas de Geometria, Topologia e Densidade de *Manifolds*** – Foram utilizadas nesse trabalho para mapear apenas um caso, quando se utiliza o parâmetro número de execuções dos algoritmos. Para testes inicias, a métrica de não-linearidade do classificador 1-NN (N4) mapeou o número de execuções, porque existe uma relação direta do erro de classificação do algoritmo de RBC e o erro retornado pelo classificador 1-NN, que nada mais é que N4.

Com a fundamentação das métricas e do mapeamento para os componentes realizada, pode-se explicar como as métricas foram utilizadas dentro do processo de evolução do AG-M.

A representação do indivíduo foi modificada a fim de considerar as métricas de complexidade como parte das tomadas de decisão do AG. Cada vez que a adaptação que lida com as métricas é utilizada, o indivíduo ganha quatro posições para tratar as métricas como parte de genótipo, que são os genes (posições do cromossomo): *Métricas*, *Operador*, *Limiar Aleatório* e *Componente*. Os mesmos passam a fazer parte da representação de todos os indivíduos.

A Figura 3.7 mostra o novo indivíduo para a adaptação proposta para o AG, que considera as quatro posições mencionadas anteriormente, e onde as demais posições podem assumir valores específicos, como mostrado na Figura 3.3.

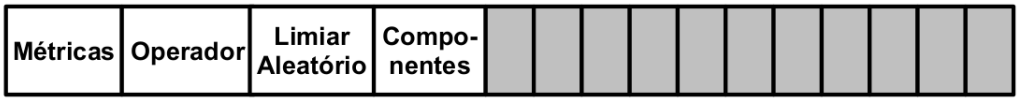


Figura 3.7: Nova representação do indivíduo para adaptar o AG.

O papel da métrica de complexidade na adaptação é criar uma condicional no indivíduo. Por exemplo, suponha que a busca gulosa do K2 foi mapeada como método de busca a partir de um vetor real representando o indivíduo. Quando o componente de pontuação local é selecionado, opta-se pela mudança do fluxo de processamento com a inserção da métrica N1.

Com isso, os genes *Métricas*, *Operador*, *Limiar Aleatório* e *Componente* vão ser utilizados. O primeiro deles estabelece que a medida de complexidade N1 está relacionada com a métrica de pontuação local. O segundo gene, o *Operador*, possui dois

alelos possíveis: a operação *Maior ou Igual que* e a operação *Menor ou igual que*. Essas operações conduzem o algoritmo durante a escolha de que qual decisão tomar.

O gene *Limiar Aleatório* é utilizado para criar o limiar para a tomada de decisão entre as opções dos alelos do gene *Componentes*. Assim, o valor encontrado na posição do *Limiar Aleatório* do vetor real, que representa o indivíduo, é multiplicado pelo valor máximo da métrica de complexidade, nesse caso, N1. Suponha que o valor encontrado é $0,345$ e o valor máximo dessa métrica é $1,0$. Então $0,345$ se mantém. Caso esse valor seja *Maior ou Igual que* o valor encontrado dentro do arquivo de métricas (CSV) para a base de dados passada como entrada, a métrica de pontuação BAYES, por exemplo, é escolhida. Caso contrário, decide-se utilizar uma das duas métricas de pontuação (MDL ou AIC), onde cada métrica possui a mesma chance de ser escolhida.

Para os demais componentes que podem estar lidados às métricas de complexidade, o funcionamento é similar. O que muda, são as métricas de complexidade e seus intervalos.

Capítulo 4

Resultados e Discussões

Este capítulo apresenta os resultados da evolução de Redes Bayesianas de Classificação e está dividido em três partes principais: testes do método proposto em execuções direcionadas a bases de dados específicas, testes em execuções direcionadas a conjuntos de bases dados semelhantes e testes com conjuntos bases de treino e teste distintos.

Para a primeira parte dos experimentos, o objetivo é personalizar algoritmos de RBCs para bases específicas, considerando como mais apropriado aquele que obteve um maior desempenho de classificação durante o treinamento do AG.

Já para segunda parte, em que existem conjuntos de bases de diferentes contextos, o propósito de evoluir os algoritmos de RBCs é definir algoritmos para diferentes tipos de grupos de bases com características semelhantes. É importante ressaltar que definir quais bases são semelhantes é ainda um problema em aberto na literatura. Aqui, utilizou-se características dos dados para agrupá-las.

Por fim, o propósito da terceira parte é analisar a geração de algoritmos em que conjuntos de bases de dados de treino e teste possuem pouca semelhança. Esse teste possui também o objetivo de verificar a generalidade do método proposto, visto que os dados de teste mudam de suas características em relação ao treinamento. A mudança de característica é determinada pelo agrupamento realizado sobre as bases de dados.

No total, 20 bases de dados do repositório da UCI (University of California Irvine) [Asuncion & Newman, 2007] foram selecionadas para validação do presente trabalho. Na primeira parte, em que algoritmos personalizados bases de dados específicas, 15 delas foram utilizadas, conforme descrito na Seção 4.2.

Nas segunda e terceira partes dos testes, o foco do método foi dado a conjuntos de bases de dados, e as 20 bases foram agrupadas de acordo com suas particularidades. Para o agrupamento, características das bases e métricas de complexidade de dados foram calculadas e utilizadas. Na segunda parte, bases do mesmo grupo foram consi-

deradas para treino e teste. Para complementar a validação e verificar a importância de grupos de bases, a terceira parte dos testes considera também bases de diferentes grupos para treino e teste, e uma divisão aleatória das bases para treinar e testar a solução apresentada pelo algoritmo genético. Durante a Seção 4.3, serão discutidos mais detalhes sobre o processo de agrupamento e experimentos sobre conjuntos de bases de dados.

4.1 Planejamento Experimental

Esta seção tem por objetivo definir o planejamento experimental seguido durante a execução das três partes dos experimentos discutidas anteriormente.

Dada a natureza do método, dois tipos de comparações foram feitas: comparação do AG com um método local baseado em uma busca gulosa (BG) e comparação dos algoritmos gerados pelo AG com outros algoritmos estado-da-arte de RBCs.

A comparação com uma busca local mostra se o AG é realmente um bom método para evoluir algoritmos de RBCs. Além disso, a BG foi adotada a fim de contrastar abordagens locais (a própria BG) e abordagens globais (algoritmo genético para evoluir RBCs). A BG é um método simples que executa uma busca local sobre o espaço de componentes de RBCs. O Algoritmo 1 exemplifica o seu funcionamento. Um conjunto de p soluções são geradas aleatoriamente e representadas por um vetor real (da mesma forma que os indivíduos na primeira população do AG, ver a Seção 3.3). O parâmetro p do método define o feixe de soluções do método, ou seja, a quantidade de melhores soluções considerada a cada momento da busca. Dadas essas soluções, as mesmas são mapeadas para algoritmos e apresentadas a um conjunto de treino e validação, onde define-se as p melhores soluções atuais.

O processo de busca se inicia e, para cada posição do vetor, os seus possíveis valores são variados com a finalidade de encontrar os sucessores e verificar se algum deles possui uma solução melhor que a atual. Para a primeira posição do indivíduo, todos os possíveis valores daquele componente são testados, e se algum deles melhorar a qualidade de alguma solução atual, essa é atualizada. Ressalta-se que a solução inicial, gerada aleatoriamente, não é testada novamente e somente é substituída caso haja uma outra solução melhor para aquele conjunto de treinamento. Nos próximos passos, o mesmo procedimento é aplicado para as próximas posições, mantendo os valores dos componentes já pesquisados como os melhores valores encontrados pela BG. O valor de p adotado foi igual a um (1) e a BG possui um limite de complexidade superior de $O(1100 \cdot p)$ soluções geradas e avaliadas pelo método.

Algoritmo 1: Busca Gulosa (p)

```
// p : tamanho do feixe de soluções

// Inicialização do conjunto de melhores algoritmos de RBC:
atuais ← p soluções iniciais.
Avalie cada solução inicial de acordo com base(s) de treino e validação
melhores ← ∅
para cada solução s em atuais faça
  pos ← 1
  enquanto pos ≤ |solução| faça
    // Adiciona a solução atual e seus sucessores ao conjunto auxiliar de
    // algoritmos de RBC:
    conjunto ← ∅
    conjunto ← conjunto ∪ s
    para cada sucessor s' de s na posição pos faça
      Avalie sucessor s' de acordo com base(s) de treino e validação
      conjunto ← conjunto ∪ s'
    fim para
    // Seleciona apenas as melhores soluções para a próxima iteração, visto
    // os conjuntos de treinamento e validação:
    m ← elemento de conjunto com melhor medida F da avaliação
    se |melhores| < p então
      melhores ← melhores ∪ m
    senão
      se |melhores| = p então
        Substitua o elemento e com menor medida F de melhores
        por m caso: F_e < F_m
      fim se
    fim se
    pos + +
  fim enquanto
  atuais ← atuais − {s}
fim para
retorna melhores
```

Para este trabalho, os algoritmos estado-da-arte escolhidos foram o Naïve Bayes (NB), o Tree Augmented Naïve Bayes (TAN) e o K2. A Subseção 2.1.3 expõe mais detalhes sobre esses três algoritmos de RBCs. Contudo, uma breve descrição e justificativa do uso dos mesmos o e seus parâmetros associados (adotados) será feita a seguir.

O Naïve Bayes supõe que os nós atributos são independentes dado o nó classe e, portanto, possui sempre uma busca fixa onde somente os parâmetros da RBC são

aprendidos. Para o TAN, uma abordagem baseada em restrições é utilizada para realizar a busca e assume que os atributos previsores se relacionam na forma de um árvore. Já o K2 realiza a busca por meio de uma abordagem gulosa baseada em Busca e Pontuação determinando as relações entre atributos de uma base por um grafo acíclico direcionado. A razão da escolha desses três algoritmos está no fato de cada um deles tratar as relações entre os atributos das bases de dados de forma diferente.

No K2, a métrica de pontuação Bayesiana foi utilizada para medir a qualidade da RBC durante o processo de busca e o número máximo de pais, que precedem na ordenação causal, possui para cada nó da RBC um valor máximo de dois outros nós. Para todos os algoritmos, o valor do alfa do estimador foi fixado em $0,5$, que é um valor padrão no *framework* WEKA.

É importante frisar que, por serem métodos estocásticos, todos os testes referentes ao AG e à BG correspondem a cinco (5) execuções com diferentes sementes pseudo-aleatórias. Além disso, para cada teste uma validação cruzada de cinco partições foi executada, retornando como resultado a medida F (apropriadamente definida na Seção 3.4) e um algoritmo associado.

A fim de comparar estatisticamente os métodos e determinar a não-aleatoriedade dos resultados obtidos, a abordagem proposta por Demšar [2006] foi seguida. Essa abordagem busca comparar vários algoritmos em várias bases de dados ou em diversos grupos de bases de dados, baseando-se no uso do Teste de Friedman com um teste *post-hoc* correspondente. O Teste de Friedman é uma contraparte não-paramétrica do Teste ANOVA e é um teste estatístico de modelo livre de distribuição de probabilidades. Sendo assim, seja R_j a soma dos postos (*rankings*) da coluna j (classificador j), dentre os k classificadores, e N o número de bases de dados (ou o número de grupos de bases de dados), o Teste de Friedman compara os k classificares atribuindo postos para cada classificador j em cada base (grupos de bases) i dentre as (os) N possíveis. Os postos atribuídos estão relacionados com as médias das execuções dos algoritmos.

O χ_F^2 de Friedman [Friedman, 1937] com essas definições é dado por:

$$\chi_F^2 = \left[\frac{12}{Nk(k+1)} \cdot \sum_{j=1}^{k} R_j \right] - 3N(k+1), \tag{4.1}$$

e é distribuído de acordo com o χ_F^2 com $k-1$ graus de liberdade, quando N e k são grandes o suficiente.

Iman & Davenport [1980] mostrou que o χ_F^2 de Friedman é indesejavelmente conservador e derivou um ajuste:

$$F_f = \frac{(N-1)\cdot\chi_F^2}{N\cdot(k-1)-\chi_F^2}, \tag{4.2}$$

que é distribuído de acordo com a *Distribuição F* com $k-1$ e $(k-1)(N-1)$ graus de liberdade.

Se a hipótese nula de similaridade de desempenhos de classificação é rejeitada, então procede-se com o Teste *post-hoc* de Nemenyi para comparações par-a-par. Os desempenhos de dois classificadores são significativamente diferentes se seus postos médios diferirem em pelo menos a diferença crítica (DC):

$$DC = q_\alpha\sqrt{\frac{k(k+1)}{6N}}, \tag{4.3}$$

onde valores críticos q_α são baseados na estatística de Student dividida por $\sqrt{2}$.

4.2 Geração de Algoritmos de RBCs Personalizados

Esta seção mostra os resultados da evolução de RBCs considerando uma base de dados específica. Neste caso, o algoritmo genético empregado para evolução aprende o modelo personalizado a partir de uma parte dos dados durante a fase de treinamento e verifica a corretude do modelo resultante na fase de teste.

Como mencionado anteriormente, 15 bases foram utilizadas e selecionadas nesses testes devido a diferenças em suas características, como no tipo e no número de atributos e na quantidade de instâncias.

A Tabela 4.1 apresenta as bases de dados e suas principais características, incluindo número de instâncias, atributos e classes.

4.2.1 Ajuste dos Parâmetros do AG

Primeiramente, foi feita uma análise sobre os parâmetros do algoritmo genético para verificar se existe alguma configuração do mesmo que possa melhorar estatisticamente os resultados frente as bases de dados. Como o número de parâmetros é grande, foram fixados valores para quatro deles: tamanho do torneio, número de gerações, tamanho da população e quantidade de gerações para aleatorizar os dados de treinamento. Os valores fixados são os seguintes:

- Torneio com valor de K (tamanho) igual a dois (valor padrão da literatura);
- População com tamanho 35 com elitismo de tamanho igual a um;

Tabela 4.1: Bases de dados utilizadas e suas características.

Bases de Dados	#Instâncias	#Atributos	Tipo dos Atributos	#Classes	Faltantes?
Balance Scale	625	4	Inteiro	3	Não
Breast Cancer (W)	286	9	Categórico	2	Sim
Car	1.728	6	Categórico	4	Não
CMC	1.473	9	Booleano/Inteiro/Categórico	3	Não
Credit (A)	690	14	Booleano/Real/Categórico	2	Sim
Diabetes	768	8	Real	2	Não
Ecoli	336	7	Real	8	Não
Glass	214	9	Real	7	Não
Haberman	306	4	Inteiro	2	Não
Heart (C)	303	12	Booleano/Inteiro/Real/Categórico	3	Sim
Iris	150	5	Real	3	Não
Led Diplay Domain	2.880	7	Booleano	10	Não
Liver Disorders	345	7	Inteiro/Real	2	Não
Monks	432	6	Booleano/Categórico	2	Não
Tic Tac Toe	958	10	Inteiro	2	Não

- Número de gerações igual a 35. A cada cinco (5) gerações, os dados de treinamento são reamostrados para evitar *overfitting*.

Os valores de número de gerações e tamanho da população foram determinados em testes preliminares, de acordo com o tempo computacional dispendido pelo método.

Já os parâmetros ajustados foram a de taxa de mutação e a taxa de cruzamento. A Tabela 4.2 mostra a variação da taxa de mutação (M) e de cruzamento (C), seguido dos resultados para cada base de dados para o algoritmo genético. Ressalta-se que todos os resultados reportados correspondem á média (com desvio-padrão) de cinco (5) execuções do AG utilizando validação cruzada de cinco (5) partições (25 execuções).

Tabela 4.2: Ajuste de parâmetros de cruzamento (C) e mutação (M).

Bases	C/M=0,6/0,4	C/M=0,7/0,3	C/M=0,8/0,2	C/M=0,9/0,1
Balance Scale	0,695 (0,108)	0,716 (0,039)	0,715 (0,041)	0,715 (0,046)
Breast Cancer (W)	0,698 (0,079)	0,718 (0,064)	0,711 (0,075)	0,701 (0,066)
Car	0,943 (0,025)	0,949 (0,026)	0,945 (0,025)	0,930 (0,086)
CMC	0,492 (0,043)	0,503 (0,040)	0,497 (0,041)	0,499 (0,041)
Credit (A)	0,859 (0,024)	0,860 (0,016)	0,857 (0,021)	0,853 (0,021)
Diabetes	0,745 (0,034)	0,739 (0,028)	0,723 (0,061)	0,733 (0,050)
Ecoli	0,811 (0,029)	0,780 (0,051)	0,798 (0,038)	0,777 (0,045)
Glass	0,628 (0,102)	0,620 (0,092)	0,623 (0,098)	0,599 (0,101)
Haberman	0,665 (0,068)	0,648 (0,086)	0,665 (0,082)	0,642 (0,087)
Heart (C)	0,829 (0,033)	0,820 (0,032)	0,828 (0,029)	0,823 (0,032)
Iris	0,935 (0,031)	0,883 (0,017)	0,928 (0,035)	0,921 (0,050)
Led Diplay Domain	0,733 (0,025)	0,708 (0,146)	0,707 (0,146)	0,735 (0,022)
Liver Disorders	0,447 (0,103)	0,446 (0,103)	0,447 (0,103)	0,447 (0,103)
Monks	0,346 (0,051)	0,336 (0,072)	0,338 (0,055)	0,352 (0,039)
Tic Tac Toe	0,700 (0,020)	0,685 (0,054)	0,700 (0,020)	0,700 (0,020)

Seguindo os passos descritos anteriormente neste capítulo, foi realizado um teste estatístico baseado no Teste de Friedman. Os valores de estatísticas desse teste estão

presentes na Tabela 4.3.

Tabela 4.3: Teste de Friedman e seu ajuste para comparar parametrizações do AG.

Estatísticas	**C/M=0,6/0,4**	**C=0,7/0,3**	**C/M=0,8/0,2**	**C/M=0,9/0,1**
Soma dos postos	43	36	39	32
Média dos postos	2,867	2,400	2,600	2,133
Média dos valores	0,702	0,694	0,699	0,695
Desvio-padrão	0,171	0,167	0,171	0,166
χ_F^2	2,600			
F_F	0,924			

O valor crítico de $F(k-1;(k-1)(N-1)) = F(3;42)$ para $\alpha = 0,05$ é $2,827$. Desde que $F_F < F_{0,05}(3;42)$, a hipótese nula de similaridade entre os classificadores é aceita, não necessitando do Teste de *post-hoc* de Nemenyi para encontrar qual método provê resultados superiores.

Dada a similaridade da variação dos parâmetros do algoritmo genético, escolheu-se para próximos testes aquele com taxas de mutação e cruzamento iguais a $0,1$ e $0,9$, respectivamente.

4.2.2 Comparações entre Algoritmos Genéticos, Algoritmos Estado-da-Arte e Busca Gulosa

Com o ajuste dos parâmetros realizado e averiguada a similaridade entre a variação dos parâmetros de mutação e cruzamento, os próximos testes seguirão os valores especificados na subseção anterior.

Sendo assim, esta subseção tem por objetivo comparar versões do algoritmo genético (AG) para evolução de RBCs com uma busca gulosa e com três algoritmos considerados estado-da-arte (Naïve Bayes, TAN e K2).

Para fins de comparações, duas versões do algoritmo genético para evolução de Redes Bayesianas de Classificação são definidas:

1. **Algoritmo Genético (AG):** Seu processo completo de funcionamento está fundamentado na Seção 3.1.

2. **Algoritmo Genético Inicializado (AG-B):** Introduz uma simples modificação em relação ao AG. Nela, são incluídos dentro da população inicial do AG três algoritmos de RBCs popularmente conhecidos. O objetivo é verificar se o AG-B é capaz de melhorar com os algoritmos de tradicionais mais rapidamente.

A Tabela 4.4 apresenta os resultados das duas versões do AG, da BG e dos métodos de comparação para as 15 bases de dados escolhidas.

Tabela 4.4: Resultados das versões do genético, da BG e dos métodos de comparação.

Base de Dados	AG	AG-B	BG	NB	TAN	K2
Balance Scale	0,715 (0,046)	0,693 (0,101)	0,691 (0,109)	**0,719** (0,048)	0,703 (0,055)	0,710 (0,053)
Breast Cancer	0,701 (0,066)	0,701 (0,076)	0,701 (0,090)	0,733 (0,038)	0,670 (0,053)	0,719 (0,054)
Car	0,930 (0,086)	0,972 (0,016)	0,914 (0,067)	0,849 (0,030)	0,945 (0,019)	0,906 (0,020)
CMC	0,499 (0,041)	0,492 (0,046)	0,475 (0,063)	0,504 (0,044)	0,507 (0,034)	0,493 (0,038)
Credit (A)	0,853 (0,021)	0,859 (0,025)	0,847 (0,039)	0,862 (0,036)	0,842 (0,025)	0,845 (0,017)
Diabetes	0,733 (0,050)	0,723 (0,069)	0,740 (0,028)	0,737 (0,030)	0,747 (0,020)	0,741 (0,024)
Ecoli	0,777 (0,045)	0,801 (0,022)	0,782 (0,026)	0,812 (0,014)	0,798 (0,019)	0,807 (0,014)
Glass	0,599 (0,101)	0,636 (0,122)	0,600 (0,104)	0,685 (0,040)	0,658 (0,084)	0,682 (0,034)
Haberman	0,642 (0,087)	0,664 (0,092)	0,660 (0,091)	0,679 (0,080)	0,678 (0,122)	0,679 (0,080)
Heart (C)	0,823 (0,032)	0,824 (0,031)	0,817 (0,067)	0,836 (0,038)	0,835 (0,020)	0,836 (0,024)
Iris	0,921 (0,050)	0,932 (0,052)	0,939 (0,033)	0,932 (0,030)	0,926 (0,031)	0,926 (0,031)
Led	0,735 (0,022)	0,703 (0,145)	0,732 (0,023)	0,732 (0,023)	0,735 (0,026)	0,732 (0,023)
Liver Disorders	0,447 (0,103)	0,447 (0,103)	0,442 (0,101)	0,447 (0,113)	0,447 (0,113)	0,447 (0,113)
Monks	0,352 (0,039)	0,351 (0,040)	0,344 (0,046)	0,254 (0,051)	0,337 (0,024)	0,352 (0,039)
Tic Tac Toe	0,700 (0,020)	0,692 (0,045)	0,700 (0,020)	0,700 (0,022)	0,700 (0,022)	0,700 (0,022)

Para cada linha (base de dados) e coluna (método) da Tabela 4.4, os resultados referentes às médias da métrica F com seus desvios são reportados. A separação entre todos os métodos é para destacar, de forma visual, que as versões do AG serão primeiro comparadas à BG e, em seguida, aos três métodos de comparação.

As Tabelas 4.5 e 4.6 indicam essas comparações. Para cada tabela, a abordagem proposta por Demšar [2006] será seguida com o propósito de comprovar estatisticamente os resultados, como descrito na Seção 4.1.

A Tabela 4.5 exibe o teste estatístico considerando as duas versões do algoritmo genético e a busca gulosa. O valor crítico da distribuição $F(k-1;(k-1)(N-1)) = F(2;28)$ para $\alpha = 0,05$ é $3,34$. Desde que $F_F < F_{0,05}(2;28)$, a hipótese nula de similaridade entre os classificadores é aceita, não necessitando proceder com o Teste de *post-hoc* de Nemenyi para encontrar qual método provê melhores resultados. Em outras palavras, comprova-se que os AG, AG-B e BG possuem desempenho de classificação semelhante dentro do contexto das 15 bases de dados utilizadas durante a avaliação estatística.

Tabela 4.5: Comparação entre versões do genético e BG

Estatísticas	AG	AG-B	BG
Soma dos Postos	31,000	34,500	24,500
Média dos postos	2,067	2,300	1,633
Média dos valores	0,695	0,699	0,692
Desvio-padrão	0,166	0,172	0,169
χ_F^2	3,433		
F_F	1,809		

A Tabela 4.6 apresenta o teste estatístico das duas versões do algoritmo genético

com os métodos de comparação. Com os valores do número de métodos comparados e a quantidade de cenários, o valor crítico da distribuição F de Fisher-Snedecor é $F(k-1;(k-1)(N-1)) = F(4;56)$ para $\alpha = 0,05$ é $2,537$. Dado que $F_F > F_{0,05}(4;56)$, a hipótese nula de similaridade entre os classificadores é rejeitada. Procede-se, assim, o Teste de *post-hoc* de Nemenyi para encontrar qual método provê melhores resultados. A diferença crítica (DC) é dada por:

$$DC = 2,728 \times \sqrt{\frac{5 \times 6}{6 \times 15}} = 1,575 \tag{4.4}$$

Tabela 4.6: Comparação entre versões do genético e métodos de comparação.

Estatísticas	**AG**	**AG-B**	**NB**	**TAN**	**K2**
Soma dos Postos	37,000	37,000	56,500	45,000	49,500
Média dos Postos	2,467	2,467	3,767	3,000	3,300
Média dos valores	0,695	0,699	0,699	0,702	0,705
Desvio-padrão	0,166	0,172	0,177	0,169	0,164
χ_F^2	7,480				
F_F	4,650				

Dado o valor de DC encontrado na Equação 4.4, realiza-se a diferença par-a-par entre as médias dos classificadores encontrados pelo AG, AG-B, NB, TAN e K2. A diferença das médias dos postos estão na Tabela 4.7.

Com as diferenças das médias dos postos apresentadas presentes na Tabela 4.7, pode-se confirmar a igualdade em termos de medida F entre as versões do algoritmo genético (também entre si) e os métodos de comparação.

Tabela 4.7: Diferenças entre as médias dos postos dos métodos.

Método 1	**Método 2**	**Diferença entre postos médios**
AG	NB	1,300
AG-B	NB	1,300
AG	TAN	0,533
AG-B	TAN	0,533
AG	K2	0,833
AG-B	K2	0,833

Os testes estatísticos aplicados nos resultados seguiram o trabalho de Demšar [2006], um aprimoramento do Teste de Friedman, que visa através de postos comprovar a igualdade ou a superioridade de métodos em diferentes bases de dados. Quando comparadas com os métodos a versões do algoritmo genético com a busca gulosa e, em seguida, com os métodos considerados estado-da-arte, ambas as versões (AG e AG-B)

Tabela 4.8: Comparações entre algoritmos personalizados para bases específicas.

Base de dados	Algoritmo (AG)	F (AG)	Algoritmo (AG-B)	F (AG-B)	Algoritmo (BG)	F (BG)	F (NB)	F (TAN)	F (K2)
Breast Cancer	**Repeated Hill Climbing**, métrica global SB com prior, sem prob. da classe, máximo 6 pais, descarte de atributos 6 execuções e alfa=8,008	0,716	**Busca Tabu**, métrica local Bayesiana, lista de tamanho 9, 2 execuções, descarte de atributos, máximo 2 pais e alfa=3,003	0,766	**K2**, métrica global LCKV (5 partições, 5 repetições), máximo 4 pais seleção de atributos e alfa=1,401	0,780	0,750	0,695	0,764
Car	**Repeated Hill Climbing**, métrica global LOO-CV, sem prob. da classe, cobertura de Markov máximo 4 pais, seleção de atributos, inversão de arcos, 4 execuções e alfa=1,801	0,991	**Repeated Hill Climbing**, métrica global LOO-CV, máximo 7 pais, inversão de arcos, 9 execuções e alfa=0,500	0,983	**Hill Climbing**, métrica global LOO-CV, cobertura de Markov, máximo 6 pais, sem estrutura NB inicial e alfa=0,500	0,985	0,899	0,954	0,909
Monks	**Busca Tabu**, métrica local BDeu, lista de tamanho 2, 1 execução, cobertura de Markov, máximo 3 pais e alfa=0,500	0,311	**Busca Tabu**, métrica global Cumulative-CV e sem prob. da classe, lista de tamanho 10, 5 execuções, máximo 6 pais e alfa=1,201	0,386	**GAN (Híbrido)** Complexidade das relações podem chegar a uma floresta, cobertura de Markov, métrica SB com Prior, sem prob. da classe, e alfa=0,000001	0,366	0,374	0,312	0,364

se igualaram em todas as comparações. Ao propor o algoritmo de evolução, esperava-se que, pelo menos para um subconjunto das bases de dados, os resultados do AG fossem melhores que os algoritmos tradicionais de RBC, ao menos em termos de seus parâmetros. Isso pode comprovar a robustez das RBCs para lidar com diferentes tipos de bases de dados.

A Tabela 4.8 apresenta o melhor algoritmo gerado pelos métodos para uma partição em três bases de dados. A escolha das bases está motivada pelo comportamento das medidas F associadas aos algoritmos quando comparados com as medidas F dos métodos de comparação (Tabela 4.4).

Analisando a Tabela 4.8, percebemos que os algoritmos gerados são bem distintos dos métodos utilizados para comparação (últimas três colunas). As configurações dos três métodos de comparação foram definidas na Seção 4.1. Além disso, as versões do algoritmo genético e a busca gulosa podem superar os métodos de comparação em diversas partições, entretanto o maior desafio encontrado é fazer isso ocorrer em média.

Porém, como ao fim da evolução devemos escolher apenas um algoritmo para ser utilizado, comparamos as melhores soluções encontradas pelo AG (e AG-B) no conjunto de treinamento e os métodos estado-da-arte. A Tabela 4.9 apresenta as médias dos melhores resultados acompanhada por um desvio-padrão.

Tabela 4.9: Melhores soluções encontradas pelo AG e AG-B comparadas com os três métodos estado-da-arte.

Base de Dados	AG	AG-B	NB	TAN	K2
Balance Scale	0,729 (0,051)	0,724 (0,046)	0,719 (0,048)	0,703 (0,055)	0,710 (0,053)
Breast Cancer	0,730 (0,051)	0,733 (0,053)	0,733 (0,038)	0,670 (0,053)	0,719 (0,054)
Car	0,979 (0,009)	0,983 (0,009)	0,849 (0,030)	0,945 (0,019)	0,906 (0,020)
CMC	0,521 (0,038)	0,517 (0,037)	0,504 (0,044)	0,507 (0,034)	0,493 (0,038)
Credit (A)	0,875 (0,012)	0,879 (0,017)	0,862 (0,036)	0,842 (0,025)	0,845 (0,017)
Diabetes	0,754 (0,020)	0,759 (0,032)	0,737 (0,030)	0,747 (0,020)	0,741 (0,024)
Ecoli	0,809 (0,041)	0,821 (0,004)	0,812 (0,014)	0,798 (0,019)	0,807 (0,014)
Glass	0,665 (0,107)	0,724 (0,030)	0,685 (0,040)	0,658 (0,084)	0,682 (0,034)
Haberman	0,676 (0,066)	0,694 (0,095)	0,679 (0,080)	0,678 (0,122)	0,679 (0,080)
Heart (C)	0,854 (0,019)	0,848 (0,023)	0,836 (0,038)	0,835 (0,020)	0,836 (0,024)
Iris	0,955 (0,017)	0,962 (0,027)	0,932 (0,030)	0,926 (0,031)	0,926 (0,031)
Led	0,741 (0,021)	0,740 (0,026)	0,732 (0,023)	0,735 (0,026)	0,732 (0,023)
Liver Disorders	0,447 (0,113)	0,447 (0,113)	0,447 (0,113)	0,447 (0,113)	0,447 (0,113)
Monks	0,386 (0,049)	0,400 (0,021)	0,254 (0,051)	0,337 (0,024)	0,352 (0,039)
Tic Tac Toe	0,700 (0,022)	0,700 (0,022)	0,700 (0,022)	0,700 (0,022)	0,700 (0,022)

A Tabela 4.10 apresenta o teste estatístico das melhores soluções encontradas pelo AG e AG-B com os métodos de comparação. Com os valores do número de métodos comparados e a quantidade de cenários, o valor crítico da distribuição F de Fisher-Snedecor $F(k-1;(k-1)(N-1)) = F(4;56)$ para $\alpha = 0,05$ é $2,537$. Como $F_F > F_{0,05}(4;56)$, a hipótese nula de similaridade entre os classificadores é rejeitada.

O Teste de Nemenyi é efetuado para encontrar qual o métodos mais adequados dado essas bases de dados. A diferença crítica (DC) do teste também é dada por $1,575$ (ver Equação 4.4).

Tabela 4.10: Comparação entre melhores soluções encontradas pelas versões do genético e métodos de comparação.

Estatísticas	**AG**	**AG-B**	**NB**	**TAN**	**K2**
Soma dos postos	56,000	63,000	41,000	30,500	34,500
Média dos postos	3,733	4,200	2,733	2,033	2,300
Média dos valores	0,721	0,720	0,699	0,702	0,705
Desvio-padrão	0,170	0,168	0,177	0,169	0,164
χ_F^2	20,840				
F_F	7,450				

Com as diferenças das médias dos postos apresentadas presentes na Tabela 4.11, pode-se afirmar que o AG é superior, em termos de classificação, em relação ao TAN, mas não existe evidência estatística de que seja melhor ou pior que o NB e K2. Já o AG-B é superior ao TAN e K2, mas não existe diferença em relação ao NB. Como os algoritmos gerado pelo método diferem muito entre si, dado o vasto espaço de busca, a análise de algoritmos isolados é essencial, e mostra que o AG pode encontrar soluções mais que competitiva com os métodos estado-da-arte.

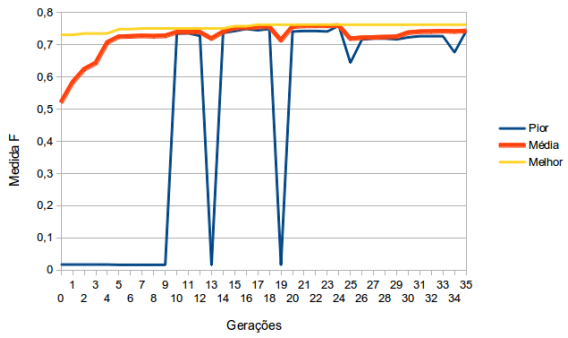
Tabela 4.11: Diferenças entre as médias dos postos dos métodos.

Método 1	**Método 2**	**Diferença entre postos médios**
AG	NB	1,000
AG-B	NB	1,467
AG	TAN	1,700
AG-B	TAN	2,167
AG	K2	1,433
AG-B	K2	1,900

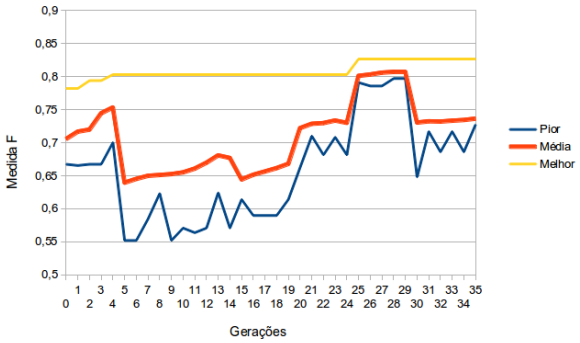
Para complementar as explicações sobre os algoritmos de RBC gerados pelo algoritmo genético, os gráficos da Figura 4.1 (a-d) ilustram a convergência de *fitness* do AG e do AG-B pela medida F (eixo y) durante as 35 gerações (eixo x) para uma partição. As curvas de evolução de ambas figuras definem as direções que tomam o melhor indivíduo, o pior indivíduo e a média de *fitness* da população com o passar das gerações. Note que os valores de *fitness* podem diminuir de uma geração para outra, mesmo utilizando elitismo, devido à reamostragem do conjunto de treinamento. Contudo, os valores de *fitness* são rapidamente retomados.

Além disso, percebe-se uma maior suavidade das curvas quando o AG-B é utilizado (Figuras 4.1c e 4.1d). A explicação encontrada é simples, já que a adição dos

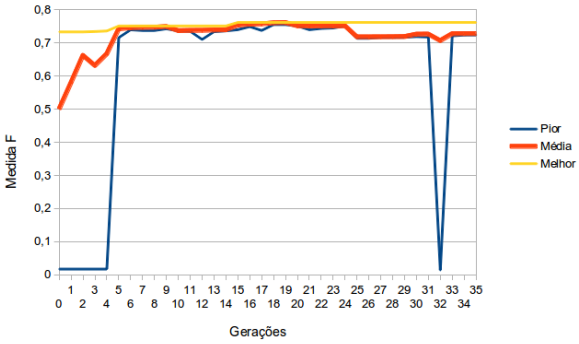
métodos na população inicial, os quais possuem na maioria das vezes um desempenho de classificação superior, permite mudar o processo de evolução. Durante a execução do AG-B, a aplicação do elitismo e dos operadores faz com que o mesmo gere algoritmos que mantenham mais estáveis as curvas de *fitness*, mesmo com as reamostragens.



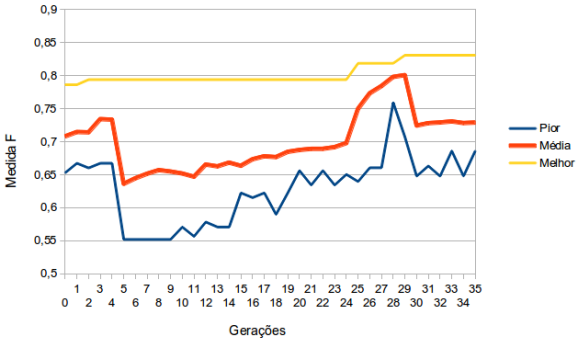
(a) Evolução do AG p/ base Led.



(b) Evolução do AG p/ base Breast(C).



(c) Evolução do AG-B p/ base Led.



(d) Evolução do AG-B p/ base Breast(C).

Figura 4.1: Curvas de evolução do AG e AG-B para duas bases de dados.

Note também que o intervalo dos valores de *fitness* que os algoritmos genéticos podem obter não é grande para a base *Breast Cancer* (Figura 4.1b e 4.1d). Esse comportamento pode ocorrer e ocorre, de fato, para outras bases de dados de outros contextos. Isso é justificado porque os algoritmos genéticos não usam quaisquer tipos de componentes que não possam fazer sentido para apenas combiná-los e gerar algoritmos com mal desempenho de classificação. Talvez um método de busca local fosse útil para explorar o espaço de busca de componentes depois que uma exploração geral inicial (algoritmos genéticos) seja executada.

A Tabela 4.12 distingue os tempos médios de execução das versões do algoritmo genético e da busca gulosa. Como o tempo de execução dos métodos de comparação

são bem inferiores aos demais, decidiu-se não monitorar os seus tempos médios de execução. Como esperado, percebe-se pela tabela que as bases que possuem maior custo computacional para execução são aquelas que possuem maior quantidade de atributos, classes ou instâncias: *Breast Cancer*, *Car*, *CMC*, *Credit (A)*, *Heart (C)* e *Led*.

Tabela 4.12: Tempos médios de execução (segundos) para testes nas bases específicas.

Base de Dados	**AG**	**AG-B**	**Guloso**
Balance Scale	329,700	959,000	52,000
Breast Cancer	5500,000	2051,000	41,000
Car	8379,000	4120,000	261,000
CMC	8534,000	5364,000	183,000
Credit (A)	23383,667	8844,000	97,000
Diabetes	979,000	621,000	14,000
Ecoli	1245,000	640,000	29,000
Glass	1193,000	630,000	11,000
Haberman	220,000	15,000	2,000
Heart (C)	11285,000	3437,000	1914,000
Iris	376,000	148,000	2,000
Led	8806,000	7646,000	3162,000
Liver Disorders	330,000	42,000	16,000
Monks	225,000	106,000	4,000
Tic Tac Toe	1206,000	327,000	254,000

4.3 Geração de Algoritmos para Conjuntos de Bases de Dados

Esta seção foca na aplicação da abordagem de evolução de RBCs considerando conjuntos de bases de dados no treinamento e no teste, ao invés de apenas uma base de dados específica. Em outras palavras, existem várias bases de treinamento e várias bases de testes, distintas umas das outras. Deste modo, o AG aprende o modelo a partir de um conjunto de bases de dados durante a fase de treinamento e verifica na fase de teste a corretude do mesmo a partir de um outro conjunto de bases de dados.

A Tabela 4.13 apresenta as 20 bases de dados e suas principais características, incluindo número de instâncias, atributos e classes. Para os experimentos desta seção, as bases de dados foram agrupadas para gerar diferentes cenários de teste e para isso, uma maior quantidade de bases de dados foi selecionada para distinção das mesmas (agrupamento de acordo com suas particularidades) e para separação em diferentes conjuntos de treinamento e teste.

Tabela 4.13: Bases de dados para formação dos grupos.

Bases de Dados	**#Instâncias**	**#Atributos**	**Tipos dos Atributos**	**#Classes**	**Faltantes?**
Balance Scale	625	4	Inteiro	3	Não
Breast Cancer (W)	286	9	Categórico	2	Sim
Car	1.728	6	Categórico	4	Não
CMC	1.473	9	Booleano/Inteiro/Categórico	3	Não
Credit (A)	690	14	Booleano/Real/Categórico	2	Sim
Credit (G)	1.000	20	Booleano/Real/Categórico	2	Não
Diabetes	768	8	Real	2	Não
Ecoli	336	7	Real	8	Não
Glass	214	9	Real	7	Não
Haberman	306	4	Inteiro	2	Não
Heart (C)	303	12	Booleano/Inteiro/Real/Categórico	3	Sim
Ionosphere	351	34	Real	2	Não
Iris	150	4	Real	3	Não
Led Diplay Domain	2.880	7	Booleano	10	Não
Monks	432	6	Booleano/Categórico	2	Não
Liver Disorders	345	7	Inteiro/Real	2	Não
Nursery	12.960	8	Categórico	5	Não
Sick	3.772	29	Booleano/Inteiro/Real/Categórico	2	Sim
Soybean	683	35	Booleano/Categórico	19	Sim
Tic Tac Toe	958	10	Inteiro	2	Não

Para o agrupamento das bases, dois tipos de informações foram utilizadas: características do tamanho, da quantidade de atributos e do número de classes (referentes à Tabela 4.13) e suas métricas de complexidade de dados previamente calculadas (informadas na Tabela 4.14 e detalhadas na Seção 3.6 do capítulo anterior).

Após serem descritas por um conjunto de 17 atributos, utilizou-se o *X-Means* [Pelleg & Moore, 2000] para agrupá-las. O *X-Means* busca o número K de grupos em uma

Tabela 4.14: Métricas utilizadas para formação dos grupos de bases e para guiar versões do algoritmo genético proposto.

Base de Dados	F1	F1v	F2	F3	F4	L1	L2	L3	N1	N2	N3	N4	T1	T2
Balance Scale	0,204	-1,000	3,000	0,000	0,000	-1,000	-1,000	-1,000	0,317	0,677	0,208	0,329	0,914	156,250
Breast Cancer (W)	0,021	0,852	0,500	0,003	0,007	0,588	0,297	0,500	0,437	0,865	0,357	0,416	1,000	31,778
Car	0,006	-1,000	0,389	0,667	1,704	-1,000	-1,000	-1,000	0,304	0,902	0,245	0,730	1,000	288,000
CMC	0,045	-1,000	2,298	0,068	0,084	-1,000	-1,000	-1,000	0,686	1,033	0,539	0,540	1,000	163,667
Credit (A)	0,364	3,703	0,001	0,032	0,068	0,290	0,145	0,143	0,546	0,182	0,335	0,354	0,987	46,000
Credit (G)	0,107	1,445	0,662	0,014	0,024	0,685	0,297	0,499	0,587	0,879	0,386	0,451	0,991	50,000
Diabetes	0,576	1,911	0,252	0,007	0,022	0,689	0,350	0,500	0,467	0,828	0,320	0,321	1,000	96,000
Ecoli	3,871	-1,000	0,191	1,000	1,000	-1,000	-1,000	-1,000	0,274	0,682	0,188	0,283	0,976	48,000
Glass	1,576	-1,000	6,013	1,000	1,000	-1,000	-1,000	-1,000	0,444	0,611	0,266	-1,000	0,995	23,778
Haberman	1,515	0,502	0,718	0,029	0,033	0,530	0,265	0,500	0,480	0,852	0,337	0,415	0,987	102,000
Heart (C)	0,276	-1,000	9,209	1,000	1,000	-1,000	-1,000	-1,000	0,591	0,905	0,413	-1,000	0,997	23,308
Ionosphere	0,614	7,040	0,000	0,191	0,994	0,455	0,117	0,142	0,234	0,635	0,134	0,172	0,946	10,324
Iris	16,041	-1,000	0,054	0,573	0,573	-1,000	-1,000	-1,000	0,087	0,200	0,040	0,020	0,893	37,500
Led Diplay Domain	1,604	-1,000	45,000	0,000	0,000	-1,000	-1,000	-1,000	0,354	1,242	0,337	0,335	1,000	411,429
Monks	0,055	0,630	0,073	0,032	0,107	0,841	0,420	0,500	0,568	0,926	0,377	0,361	1,000	57,500
Liver Disorders	-1,000	0,000	1,000	0,000	0,000	1,000	0,500	0,500	1,000	2,000	1,000	0,500	1,000	72,000
Nursery	0,004	-1,000	2,125	0,800	0,800	-1,000	-1,000	-1,000	0,358	1,000	0,232	0,394	1,000	1620,000
Sick	1,826	4,936	0,000	0,138	0,300	0,146	0,061	0,500	0,140	0,365	0,094	0,407	0,968	130,069
Soybean	1,324	-1,000	7,278	1,000	1,000	-1,000	-1,000	-1,000	0,211	0,437	0,092	0,070	1,000	19,514
Tic Tac Toe	0,074	0,292	1,000	0,000	0,000	0,693	0,347	0,500	0,482	0,978	0,309	0,277	1,000	106,444

Tabela 4.15: Dois grupos para as 20 bases de dados.

Bases de Dados	Grupo
Balance Scale, Car, CMC, Ecoli, Glass, Heart (C), Iris, Led Display Domain, Nursery, Soybean	0
Breast Cancer (W), Credit (A), Credit (G), Diabetes, Haberman, Ionosphere, Monks, Liver Disorders, Sick,Tic Tac Toe	1

faixa pré-definida. Neste trabalho, a faixa de dois (2) a 10 foi delimitada. O valor de K é definido pelo *X-Means* pela execução de divisões binárias recursivas do espaço até que se alcance o melhor valor de K dentro da faixa de valores estipulados. Para qualificar se uma divisão obtém um resultado superior ao existente, uma pontuação de uma função BIC (*Bayesian Information Criterion*) é utilizada.

As características e as métricas de complexidade de dados foram passadas como entrada para o *X-Means* e o mesmo retornou a divisão dos grupos da Tabela 4.15. Como pode ser visto, o *X-Means* separou as bases de dados em dois grupos e cada grupo possui 50% das mesmas.

Dado o agrupamento realizado pelo *X-Means*, dois tipos testes foram empregados a fim de avaliar a validade do método proposto, que são: **(i)** testes em execuções direcionadas a conjuntos de bases dados semelhantes e **(ii)** testes com conjuntos bases de treino e teste distintos. As duas subdivisões dos testes do presente capítulo serão detalhadas a seguir nas Subseções 4.3.1 e 4.3.2.

Para ambas subdivisões, assumiu-se nos experimentos os mesmos parâmetros adotados na para o AG da Seção 4.2, que lida com bases específicas, uma vez que os testes não mostraram diferença estatisticamente significativa.

Além disso, do mesmo modo que o ajuste dos parâmetros, manteve-se as comparações, de forma isolada, das versões do algoritmo genético com uma busca gulosa e com três métodos estado-da-arte (Naïve Bayes, TAN e K2). As justificativas para as escolhas desses algoritmos em comparações com as versões do algoritmo genético estão presentes na Seção 4.1.

A diferença para a seção anterior é que nesta existem quatro versões do algoritmo genético proposto para evolução de RBCs. Duas delas são referentes às versões já apresentadas na seção que aborda bases específicas, a AG e a AG-B. Aparecem nesta seção duas novas versões: AG-M e AG-MB. A versão AG-M é uma adaptação da versão AG e possui o propósito de trabalhar com métricas de complexidade de bases de dados. Tais métricas são utilizadas para guiar o algoritmo genético frente a conjuntos de bases de dados com o intuito de encontrar soluções semelhantes para tipos de dados com particularidades similares. A quarta versão também inclui os métodos de comparação dentro da população inicial de sua versão original (AG-M).

Adicionalmente, a distinção para as execuções do algoritmo genético com uma base de dados para conjuntos bases está em como se avalia um algoritmo gerado pelo AG. Enquanto que para uma base o resultado do AG é a avaliação (medida F) para somente aquela base, para um conjunto de bases a média dos resultados da avaliação é calculada.

Para os resultados para grupos de bases, outra questão em relação aos resulta-

dos para bases específicas deve ser evidenciada. No treinamento do AG, a validação cruzada não é utilizada já que se tem várias bases de dados. No teste tanto o algoritmo de RBC quanto os métodos de comparação rodam uma validação cruzada com cinco (5) partições para cada base de teste. Para cada partição, obtém-se um resultado da medida F média para as bases de teste. Cinco (5) repetições utilizando sementes pseudo-aleatórias são realizadas e para cada configuração obtém-se 25 valores da medida F como resultados de cada método (versões do AG, métodos de comparação e BG).

Esclarecidas as diferenças (e semelhanças) e estabelecidas as definições, pode-se partir para as próximas subseções para tentar analisar o restante dos testes que foram propostos no início deste capítulo.

4.3.1 Testes em Conjuntos de Bases de Dados Semelhantes

No conjunto de testes que será descrito e analisado por essa subseção, tanto o treino quanto o teste possuem bases estão no mesmo grupo, dado a Tabela 4.15 a qual apresenta os grupos formados.

O objetivo deste tipo de experimento é verificar se o algoritmo genético consegue manter os resultados dos algoritmos gerados durante o treinamento na fase de teste, visto que as características das bases de dados são ditas como similares pelo agrupamento.

Com esse objetivo em primeiro plano, configurou-se cinco (5) experimentos para validar evolução dos algoritmos de RBCs com foco em conjuntos semelhantes de bases de treino e de teste. A Tabela 4.16 delimita os experimentos, alocando as bases de dados aos respectivos conjuntos de treinamento e teste.

Dado as configurações de cada experimento, a Tabela 4.17 apresenta os resultados das quatro versões do AG, da BG e dos métodos de comparação para as cinco (5) configurações de conjuntos de bases de dados de treinamento e teste que possuem características similares.

Para cada linha dessa tabela, a identificação da configuração da primeira coluna da Tabela 4.17 é apresentada ao invés dos nomes das bases por questões de espaçamento. Cada coluna referente à essa tabela reflete os resultados execução de cada método (nomeados a partir da coluna 2). Os resultados consistem de médias da medida F juntamente com o desvio-padrão associado a validação cruzada direcionada a cada base de teste, como já especificado anteriormente.

A validação estatística para as diferentes configurações de grupos de bases é similar à seção anterior, ou seja, compara-se cada versão do algoritmo genético isoladamente

Tabela 4.16: Experimentos considerando conjuntos de bases de dados com características similares.

Configuração	Bases de treinamento	Bases de teste
1	Balance Scale, Car, CMC	Ecoli, Glass
2	Iris, Led, Glass	Car, Nursery, Soybean
3	Balance Scale, Iris, Glass, Heart (C)	Led, Car, CMC
4	Credit (A), Diabetes, Monks	Breast (C), Sick, Credit (G)
5	Breast (C), Diabetes, Tic Tac Toe, Liver Disorders	Credit (A), Ionosphere, Haberman

Tabela 4.17: Resultados das diferentes versões do algoritmo genético, *métodos de comparação* e busca gulosa para conjuntos de bases de dados no mesmo grupo.

Configuração	AG	AG-B	AG-M	AG-MB	BG	NB	TAN	K2
1	0,737 (0,059)	0,747 (0,047)	0,705 (0,063)	0,743 (0,043)	0,717 (0,072)	0,743 (0,044)	0,753 (0,041)	0,751 (0,045)
2	0,866 (0,085)	0,903 (0,089)	0,914 (0,033)	0,882 (0,063)	0,779 (0,088)	0,888 (0,011)	0,939 (0,007)	0,921 (0,009)
3	0,710 (0,015)	0,712 (0,017)	0,715 (0,023)	0,725 (0,009)	0,690 (0,017)	0,694 (0,013)	0,729 (0,007)	0,718 (0,010)
4	0,790 (0,022)	0,782 (0,030)	0,780 (0,026)	0,798 (0,021)	0,764 (0,033)	0,808 (0,014)	0,789 (0,016)	0,802 (0,021)
5	0,797 (0,049)	0,803 (0,034)	0,808 (0,037)	0,814 (0,032)	0,807 (0,035)	0,817 (0,032)	0,815 (0,032)	0,826 (0,030)

com a BG e, logo após essa primeira comparação, as versões do AG são confrontadas com os três métodos da literatura: NB, TAN e K2.

As Tabelas 4.18 e 4.20 determinam essas comparações. Para cada tabela, acompanha-se novamente o trabalho de Demšar [2006] com o objetivo de provar a validade estatística dos resultados.

A Tabela 4.18 apresenta o teste estatístico para considerando as quatro versões do algoritmo genético e a BG. Com a quantidade de métodos e cenários, pode-se concluir que o valor crítico de $F(k-1;(k-1)(N-1)) = F(4;16)$ para $\alpha = 0,05$ fica ajustado como $3,007$. Percebe-se que $F_F > F_{0,05}(4;16)$, o que faz a hipótese nula de similaridade entre os classificadores ser rejeitada. Assim, o Teste de *post-hoc* de Nemenyi é aplicado a fim de encontrar o melhor método. O uso da diferença crítica para grupos de bases de dados é semelhante ao caso de bases específicas, visto que estamos verificando a configuração dos grupos. Portanto, a DC é dada pela Equação 4.5:

$$DC = 2,728 \times \sqrt{\frac{5 \times 6}{6 \times 5}} = 2,728 \tag{4.5}$$

Tabela 4.18: Comparações entre versões do genético e a BG considerando as configurações dos grupos de bases.

Estatísticas	**AG**	**AG-B**	**AG-M**	**AG-MB**	**BG**
Soma dos postos	12	17	16	22	8
Média dos postos	2,400	3,400	3,200	4,400	1,600
Média dos valores	0,780	0,790	0,784	0,793	0,752
Desvio-padrão	0,060	0,073	0,084	0,063	0,047
χ_F^2	8,960				
F_F	3,246				

Desde que a DC foi encontrada, pode-se calcular a diferença par-a-par entre as médias dos postos dos métodos da Tabela 4.18. As diferenças estão presentes na Tabela 4.19.

Com os valores da Tabela 4.19, pode-se concluir que o AG-MB é superior à BG, quando a medida F é utilizada para comparar classificadores. Já o AG, AG-B e o AG-M são métodos equivalentes estatisticamente à BG. Quando comparadas entre si, as versões do algoritmo genético também se mostraram equivalentes.

Na Tabela 4.20, efetuou-se os testes estatísticos referentes às versões do genético contrastadas com os métodos de comparação. Como o valor crítico da distribuição $F(k-1;(k-1)(N-1)) = F(6;24)$ para $\alpha = 0,05$ é $2,508$ e $F_F > F_{0,05}(6;24)$, a hipótese de igualdade de classificadores avaliados é rejeitada. Portanto, para comparações par-

Tabela 4.19: Diferenças entre versões do genético e BG para diversas configurações de conjuntos de bases no mesmo grupo.

Método 1	**Método 2**	**Diferença entre postos médios**
AG	BG	0,800
AG-B	BG	1,800
AG-M	BG	1,600
AG-MB	BG	2,800
AG	AG-B	1,000
AG-M	AG-MB	1,200
AG	AG-M	0,800
AG	AG-MB	2,000
AG-B	AG-M	0,200
AG-B	AG-MB	1,000

a-par, a Equação 4.6 define a diferença crítica para a aplicação do Teste de *post-hoc* de Nemenyi.

$$DC = 2,949 \times \sqrt{\frac{7 \times 8}{6 \times 5}} = 4,029 \tag{4.6}$$

Tabela 4.20: Comparações entre versões do algoritmo genético e dos métodos de comparação conjuntos de bases no mesmo grupo.

Estatísticas	**AG**	**AG-B**	**AG-M**	**AG-MB**	**NB**	**TAN**	**K2**
Soma dos postos	10	16	14	20	21	29	30
Média dos postos	2,000	3,200	2,800	4,000	4,200	5,800	6,000
Média dos valores	0,780	0,790	0,784	0,793	0,790	0,805	0,803
Desvio-padrão	0,060	0,073	0,084	0,063	0,074	0,082	0,078
χ_F^2	14,314						
F_F	10,070						

Pela DC encontrada na Equação 4.6, as diferenças par-a-par entre as médias dos postos são calculadas a fim de verificar a igualdade ou a desigualdade entre os classificadores (Tabela 4.20). As diferenças par-a-par entre as médias dos postos dos métodos estão presentes na Tabela 4.21.

Dada as diferenças presentes na Tabela 4.21, pode-se perceber que todos métodos são equivalentes estatisticamente, visto que os mesmos não possuem diferenças de médias que sejam superiores à DC. Mesmo assim, percebeu-se que as versões inicializadas do algoritmo genético (AG-B e AG-MB) conseguiram melhorar seu desempenho de suas versões originais. Em todos os casos em que a versão inicializada conseguiu melhorar (diminuir) bem sua diferença em relação aos métodos de comparação.

Como feito para os testes em bases específicas, as Tabelas 4.22 e 4.23 determinam respectivamente o melhor algoritmo e a medida F gerados pelas versões do genético e pela BG para uma partição em uma configuração de cenário de conjuntos de bases de

Tabela 4.21: Diferenças entre versões do genético e métodos de comparação para diversas configurações de bases no mesmo grupo.

Método 1	**Método 2**	**Diferença entre postos médios**
AG	NB	2,200
AG	TAN	3,800
AG	K2	4,000
AG-B	NB	1,000
AG-B	TAN	2,600
AG-B	K2	2,800
AG-M	NB	1,400
AG-M	TAN	3,000
AG-M	K2	3,200
AG-MB	NB	0,200
AG-MB	TAN	1,800
AG-MB	K2	2,000

dados com características similares. É importante mencionar que as configurações dos métodos estado-da-arte foram discutidas na seção de planejamento experimental.

Percebeu-se diferenças entre soluções geradas pelas versões do genético para os métodos de comparação. A BG conseguiu gerar um método de busca igual a um dos métodos, enquanto as versões do AG não geraram soluções tão similares.

Para justificar essas questão, a Figura 4.2 (a-d) ilustra a convergência de *fitness* pela medida F (eixo y) através das 35 gerações (eixo x) das versões apresentadas do algoritmo genético para a Configuração 1. As curvas de evolução dos gráficos mostram o acompanhamento do processo iterativo das versões do genético, em que o melhor indivíduo, a média de *fitness* da população e o pior indivíduo são monitorados a cada geração. Analisa-se que quedas de *fitness* podem ocorrer geralmente na média da população e no pior indivíduo de forma mais acentuada.

A explicação para as quedas bruscas de *fitness* está na reamostragem feita a cada cinco (5) gerações. Isso faz com que os dados mudem e que soluções atuais não sejam interessantes para aqueles dados. Contudo, verificou-se o processo de adaptação do genético, que consegue manter boas soluções (curva dos melhores) quando ocorre as quedas de *fitness* e a recuperação da *fitness* média com o passar das gerações.

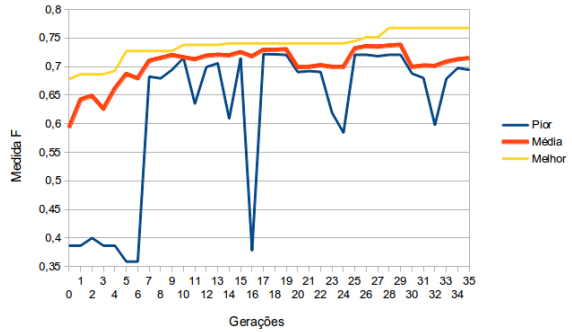
Contudo, para os mesmos gráficos da Figura 4.2 (a-d) notou-se uma reação oposta quando comparado com os testes em bases específicas (Subseção 4.2.2). A inclusão dos métodos de comparação na população inicial das duas versões originais do algoritmo genético (AG e AG-M) não estimulam a suavidade das curvas. Como podem ser visto nas subfiguras b e d dos gráficos, as curvas de piores indivíduos apresentam comportamentos menos estáveis, pois ocorrem maiores pontos de mínimo com o passar das gerações. Ou seja, durante a execução do AG-B e do AG-MB, o elitismo e os operadores fazem com que algoritmos que mantenham menos estáveis as curvas de *fitness*.

Tabela 4.22: Comparações entre algoritmos gerados para configurações de bases no mesmo grupo.

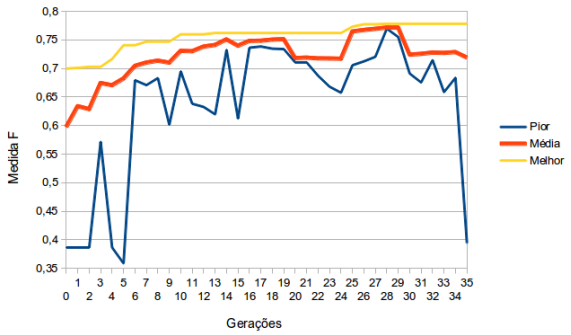
Configuração	Algoritmo (AG)	Algoritmo (AG-B)	Algoritmo (AG-M)	Algoritmo (AG-MB)	Algoritmo (BG)
1	**Repeated Hill Climbing**, métrica global LCKV e sem prob. da classe, máximo 4 pais, cobertura de Markov, inversão de arcos, 3 execuções e alfa=0,500	**Repeated Hill Climbing**, métrica global LOO-CV, máximo 9 pais, seleção de atributos, inversão de arcos, 3 execuções e alfa=0,500	**Repeated Hill Climbing**, métrica global LC, máximo 4 pais, cobertura de Markov, seleção de atributos, inversão de arcos, 9 execuções e alfa=0,500	**Repeated Hill Climbing**, métrica global LLOO, sem uso de prob. da classe, máximo 9 pais, cobertura de Markov, inversão de arcos, 8 execuções e alfa=0,500	**Naïve Bayes** e alfa=0,000001

Tabela 4.23: Medidas F para os algoritmos gerados para configurações de bases.

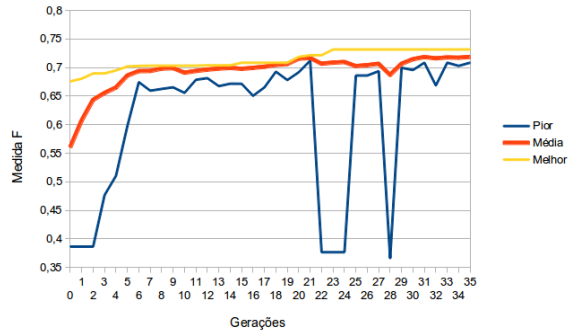
Configuração	F (AG)	F (AG-B)	F (AG-M)	F (AG-MB)	F (BG)	F (NB)	F (TAN)	F (K2)
1	0,833	0,763	0,789	0,789	0,735	0,810	0,770	0,849



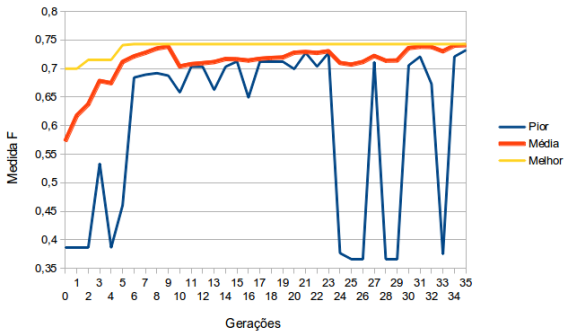
(a) Evolução do AG p/ Configuração 1.



(b) Evolução do AG-B p/ Configuração 1.



(c) Evolução do AG-M p/ Configuração 1.



(d) Evolução do AG-MB p/ Configuração 1.

Figura 4.2: Curvas de evolução do AG, AG-B, AG-M e AG-MB para Configuração 1.

Portanto, o processo evolucionário inicializado não estimula a estabilidade do sistema completo, mas direciona os algoritmos a uma maior robustez, visto que nos testes estatísticos os mesmos conseguiram igualar-se aos métodos de comparação e iguais/superiores à busca gulosa.

A Tabela 4.24 determina a média de tempos de execução das versões do algoritmo genético e da busca gulosa. As medições de tempos de execução dessa tabela demonstram que a Configuração 2 demanda maior tempo computacional e a Configuração 5 é executada em menor tempo. Esse evento também está diretamente ligado às bases de dados que compõem os grupos, em especial o de treinamento. A influência do número de instâncias, atributos e classes contribuem fortemente para o tempo de execução dos métodos.

Tabela 4.24: Tempos médios de execução (segundos) para bases no mesmo grupo.

Configuração	AG	AG-B	AG-M	AG-MB	BG
1	1479,000	3395,000	2984,000	1443,000	441,000
2	5954,000	14894,000	20527,000	7343,000	599,000
3	889,000	307,000	375,000	1903,000	182,000
4	5242,000	1207,000	1618,000	1896,000	151,000
5	653,000	964,000	830,000	1198,000	145,000

4.3.2 Testes em Conjuntos de Bases de Dados Distintas

Nesta última parte dos experimentos, os testes foram executados sobre conjuntos de bases dados de treinamento e teste possuem características distintas. Em outras palavras, o algoritmo genético não foi treinado e testado com bases com bases do mesmo grupo (ver Tabela 4.15), mas com dados pertencentes a grupos diferentes.

Para isso, os dois cenários seguintes são considerados a fim de complementar a validação do método proposto de evolução de algoritmos de RBC:

1. **Bases de treinamento e teste de ambos os grupos:** O treino e o teste possuem bases de ambos os grupos. Neste cenário, o algoritmo genético empregado para evolução de RBCs almeja construir um modelo que mantenha os resultados e que também possa generalizar em outros tipos de dados, isto é: em outros contextos. O contexto pode ter diversas definições considerando a área de mineração de dados, contudo estamos considerando o contexto como o grupo que a base se encaixa.

   A manutenção dos resultados é definida para esse cenário porque o AG possui pelo menos uma base de cada grupo no treino e no teste. Já a ideia de generalização foi almejada nesse contexto pois os conjuntos de dados de treinamento podem ser bem diferentes do teste. Por exemplo, o conjunto de treinamento pode conter duas bases do grupo 1 e duas bases do grupo 2, enquanto o teste pode ter somente uma base de cada grupo. Visto a diferença dos dados da base, o algoritmo precisa se preocupar em generalizar.

2. **Bases de treinamento e teste selecionadas aleatoriamente:** O conjunto de bases treino e o conjunto de bases de teste são escolhidos de forma aleatória. Dessa forma, não se tem certeza se existe uma base de dados de cada grupo tanto no conjunto de treino, quanto no conjunto de teste. Esse tipo de cenário é para testar situações que geralmente ocorrem e em que o algoritmo necessita lidar com a mudança de distribuição dos dados do treinamento no teste. Isto é, em situações reais, o comportamento de tratar com generalidade os dados é bem

Tabela 4.25: Experimentos considerando os dois cenários especificados.

Configuração	Bases de treinamento	Bases de teste	Cenário
1	Diabetes, Balance Scale, Monks	Soybean, Sick	1
2	Balance Scale, Iris, Credit (A), Breast (C)	Nursery, Led, Sick, Ionosphere	1
3	Car, CMC, Monks, Diabetes	Glass, Ecoli, Haberman, Credit (G)	1
4	Led, Glass, Tic Tac Toe, Credit (A)	Credit (G), Monks, Nursery, CMC	1
5	Car, Led, Breast Cancer, Monks	Balance Scale, Iris, Soybean, Diabetes	1
6	Ecoli, CMC, Diabetes, Liver Disorders	Breast Cancer, Ionosphere, centering Car, Heart (C)	1
7	Car, Liver Disorders, Tic Tac Toe	Nursery, Credit (G)	2
8	Ecoli, Led, Iris	Car, CMC, Haberman	2
9	Glass, Breast Cancer, Monks	Tic Tac Toe, Ionosphere, Liver Disorders	2
10	Balance Scale, Haberman, Liver Disorders	Credit (A), Heart (C), CMC	2

Tabela 4.26: Resultados das diferentes versões do algoritmo genético. métodos de comparação e busca gulosa para conjuntos de bases de dados em grupos distintos.

Configuração	Cenário	AG	AG-B	AG-M	AG-MB	BG	NB	TAN	K2
1	1	0,850 (0,075)	0,908 (0,075)	0,817 (0,077)	0,839 (0,079)	0,862 (0,116)	0,947 (0,009)	0,958 (0,012)	0,954 (0,011)
2	1	0,879 (0,011)	0,884 (0,011)	0,882 (0,013)	0,881 (0,012)	0,875 (0,011)	0,873 (0,009)	0,887 (0,008)	0,884 (0,008)
3	1	0,815 (0,025)	0,820 (0,020)	0,817 (0,023)	0,826 (0,022)	0,813 (0,026)	0,817 (0,027)	0,827 (0,020)	0,828 (0,022)
4	1	0,622 (0,021)	0,627 (0,018)	0,628 (0,022)	0,629 (0,017)	0,620 (0,017)	0,625 (0,017)	0,604 (0,014)	0,628 (0,016)
5	1	0,816 (0,023)	0,821 (0,019)	0,820 (0,016)	0,832 (0,015)	0,815 (0,020)	0,829 (0,015)	0,834 (0,016)	0,834 (0,015)
6	1	0,833 (0,021)	0,837 (0,018)	0,830 (0,023)	0,832 (0,024)	0,794 (0,035)	0,824 (0,024)	0,832 (0,023)	0,837 (0,022)
7	2	0,834 (0,018)	0,835 (0,016)	0,834 (0,023)	0,838 (0,017)	0,832 (0,024)	0,814 (0,016)	0,819 (0,015)	0,821 (0,014)
8	2	0,670 (0,064)	0,691 (0,027)	0,691 (0,027)	0,660 (0,073)	0,641 (0,040)	0,673 (0,026)	0,704 (0,020)	0,698 (0,024)
9	2	0,690 (0,036)	0,691 (0,037)	0,689 (0,038)	0,692 (0,036)	0,689 (0,037)	0,688 (0,037)	0,696 (0,039)	0,695 (0,038)
10	2	0,718 (0,031)	0,724 (0,027)	0,694 (0,052)	0,723 (0,023)	0,711 (0,048)	0,732 (0,023)	0,725 (0,021)	0,735 (0,023)

mais relevante. Por exemplo, em muito dos casos têm-se um pequeno conjunto de bases de treinamento que podem estar em grupos totalmente divergentes do teste.

Com os dois possíveis cenários, configurou-se 10 experimentos para averiguar o comportamento da evolução dos algoritmos de RBCs com foco em conjuntos de bases de treinamento e de teste com características distintas. A Tabela 4.25 delimita os experimentos, alocando as bases de dados e quais cenários as mesmas se encaixam.

A Tabela 4.26 mostras os resultados para todas as versões do algoritmo genético e dos métodos utilizados para comparação (estado-da-arte e busca gulosa). As duas primeiras colunas determinam a configuração e seu cenário de acordo com a Tabela 4.25. Os nomes de cada base foram omitidos por questões de espaço. A partir da terceira coluna, os resultados das médias e desvio-padrão da medida F são listados.

As Tabelas 4.27 e 4.29 especificam as comparações estatísticas [Demšar, 2006] realizadas isoladamente entre as versões do algoritmo genético, a busca gulosa e os métodos estado-da-arte da área de RBCs. De acordo com a Tabela 4.27, que compara as versões do algoritmo genético e a BG, o valor crítico de $F(k-1;(k-1)(N-1)) = F(4;36)$ para $\alpha = 0,05$ é igual a $2,634$. É visto que $F_F > F_{0,05}(4;36)$, fazendo com a hipótese nula de semelhança entre os classificadores seja rejeitada.

Tabela 4.27: Comparações entre versões do genético e a BG para conjuntos de bases em grupos distintos.

Estatísticas	**AG**	**AG-B**	**AG-M**	**AG-MB**	**BG**
Soma dos postos	27	43	27	39	14
Média dos postos	2,700	4,300	2,700	3,900	1,400
Média dos valores	0,773	0,784	0,770	0,775	0,706
Desvio-padrão	0,089	0,094	0,086	0,090	0,251
χ_F^2	20,960				
F_F	9,908				

Assim, o Teste de *post-hoc* de Nemenyi é aplicado a fim de encontrar o método mais adequado para essas bases, dado os valores da medida F. Nesse caso, faz-se uso da diferença crítica, da mesma forma que no teste anterior. Portanto, a DC é dada pela Equação 4.7:

$$DC = 2,728 \times \sqrt{\frac{5 \times 6}{6 \times 10}} = 1,925 \tag{4.7}$$

Com o valor de DC encontrado, pode-se calcular a diferença par-a-par entre as médias dos postos dos métodos da Tabela 4.27 e compará-los com a diferença crítica para realizar afirmações sobre a semelhança dos classificadores. As diferenças estão presentes na Tabela 4.28.

Tabela 4.28: Diferenças entre versões do genético e BG para diversas configurações de conjuntos de bases em grupos distintos.

Método 1	**Método 2**	**Diferença entre postos médios**
AG	BG	1,300
AG-B	BG	2,900
AG-M	BG	1,300
AG-MB	BG	2,500
AG	AG-B	1,600
AG-M	AG-MB	1,200
AG	AG-M	0,000
AG	AG-MB	1,200
AG-B	AG-M	1,600
AG-B	AG-MB	0,400

Com os valores da Tabela 4.28 e de DC (Equação 4.7), pode-se afirmar as versões inicializadas do algoritmo genético são superiores à BG, pois suas diferenças de postos foram maiores que a diferença crítica. Isso mostra que utilizar inicialização do genético com os métodos de comparação pode ser uma boa abordagem a fim de criar classificadores mais precisos. Além disso, pode-se comprovar que quando comparadas entre si, as versões do algoritmo genético se mostraram equivalentes mais uma vez.

A comparação estatística realizada na Tabela 4.29 caracteriza os testes estatísticos entre as versões do genético e dos métodos de comparação para bases de dados de grupos distintos. Calcula-se o valor crítico de $F(k-1;(k-1)(N-1)) = F(6;54)$ para $\alpha = 0,05$ e o mesmo possui valor igual a $2,272$. É visto que $F_F > F_{0,05}(6;54)$, o que consequentemente rejeita a hipótese nula.

Tabela 4.29: Comparações entre versões do algoritmo genético e dos métodos de comparação conjuntos de bases em grupos distintos.

Estatísticas	**AG**	**AG-B**	**AG-M**	**AG-MB**	**NB**	**TAN**	**K2**
Soma dos postos	26	46	29	41	27	52	59
Média dos postos	2,600	4,600	2,900	4,100	2,700	5,200	5,900
Média dos valores	0,773	0,784	0,770	0,775	0,782	0,789	0,792
Desvio-padrão	0,0891	0,0937	0,0857	0,0897	0,0999	0,1045	0,0994
χ_F^2	22,0286						
F_F	5,221						

O Teste de *post-hoc* de Nemenyi é efetuado e o valor da DC é dado ela Equação 4.8. O valor de DC pode então comparado com a diferença dos postos médios dos métodos, o que pode ser visto na Tabela 4.30.

$$DC = 2,949 \times \sqrt{\frac{7 \times 8}{6 \times 10}} = 2,849 \tag{4.8}$$

Tabela 4.30: Diferenças entre versões do genético e métodos de comparação para diversas configurações bases .

Método 1	Método 2	Diferença entre postos médios
AG	NB	0,100
AG	TAN	2,600
AG	K2	3,300
AG-B	NB	1,900
AG-B	TAN	0,600
AG-B	K2	1,300
AG-M	NB	0,200
AG-M	TAN	2,300
AG-M	K2	3,000
AG-MB	NB	1,400
AG-MB	TAN	1,100
AG-MB	K2	1,800

Dada as diferenças presentes na Tabela 4.30, pode ser analisado que as versões não-inicializadas dos métodos são inferiores ao K2. Conseguiu-se, a partir da inicialização, melhorar os resultados, pois os algoritmos de RBCs gerados mantiveram-se estatisticamente iguais à todos os métodos dados como estado-da-arte, como esperado.

Aqui, analisou-se a perda de soluções nos algoritmos genéticos inicializados (AG-B e AG-MB), pelo fato dos algoritmos evoluídos serem diferentes dos métodos de comparação. Dois fatores explicam conjuntamente essa situação ocorrida. O primeiro fator é o mesmo da seção anterior: a reamostragem do treinamento. Isso faz com muitas soluções consideradas boas em um cenário sejam perdidas com o passar das gerações do AG. Já o segundo fator é influenciado pelas bases de teste, que podem se comportar de forma completamente distinta do treinamento em um dado algoritmo de RBC. Com isso, mesmo que a reamostragem não seja realizada, o treinamento pode perder soluções interessantes, no contexto de classificação, quando treina o AG com diversas bases de treinamento completamente opostas às bases de teste. Mesmo com essas perdas, as versões inicializadas não diferiram dos métodos de comparação nos testes estatísticos.

Os testes estatísticos realizados conseguiram comprovar a eficácia do uso da inicialização da primeira população com os algoritmos estado-da-arte: NB, TAN e K2. Na Tabela 4.26, a maioria dos resultados das configurações que foram inicializados foram superiores que sua versão não-inicializada. Durante os testes estatísticos, também notou-se essa mesma diferença, pois os métodos inicializados mantiveram-se equiparáveis aos métodos de comparação da mesma forma que seus métodos não-inicializados.

As Tabelas 4.31 e 4.32 apresentam, respectivamente, o melhor algoritmo e a medida F gerados pelas versões do genético e pela BG para resultados em dois conjuntos de bases de dados presentes em grupos distintos. As configurações escolhidas estão em

Tabela 4.31: Comparações entre algoritmos gerados para configurações de grupos de bases.

Configuração	Algoritmo (AG)	Algoritmo (AG-B)	Algoritmo (AG-M)	Algoritmo (AG-MB)	Algoritmo (BG)
3	**Hill Climbing**, métrica global LLOO e sem prob. da classe, máximo 6 pais, cobertura de Markov sem estrutura NB inicial e alfa=3,904	**K2**, métrica local Bayesiana, máximo 7 pais, descarte de atributos, inversão dos arcos, cobertura de Markov e alfa=0,500	**GAN (Híbrido)**, métrica global LCKV (10 partições, 5 repetições), Complexidade das relações podem chegar a uma floresta das relações, cobertura de Markov, descarte de atributos e alfa=2,002	**Naïve Bayes** e alfa=0.500	**Naive Bayes** e alfa=0,000001
7	**Hill Climbing**, métrica global LKCV, cobertura de Markov, seleção de atributos, máximo 10 pais, sem estrutura NB inicial e alfa=0,500	**TAN**, métrica global LogC, cobertura de Markov, descarte de atributos e alfa=0,500	**Repeated Hill Climbing**, métrica global Cumulative-CV, sem prob. da classe, máximo 4 pais, inversão de arcos, 8 execuções e alfa=0,701	**TAN**, usa prior, restrições p/ arestas, métrica Bayesiana e alfa=0,500	**Hill Climbing**, métrica global LOO-CV, cobertura de Markov, máximo 5 pais, seleção de atributos, inversão de arcos, sem estrutura NB inicial e alfa=0,500

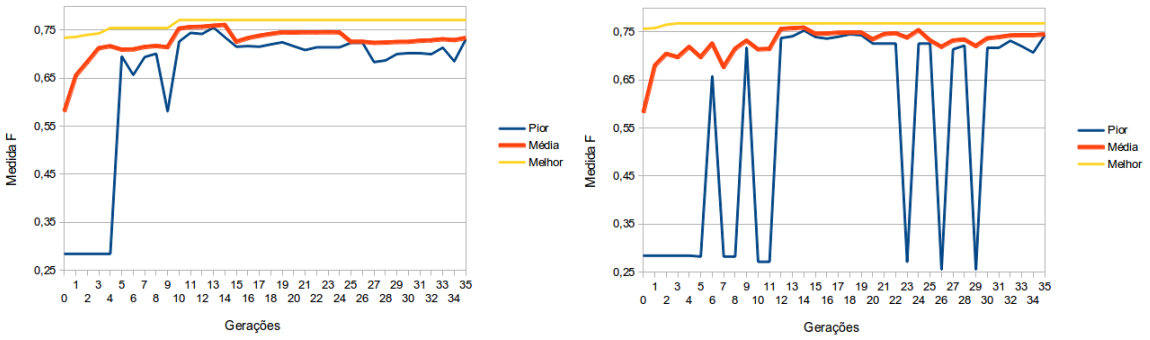
Tabela 4.32: Medidas F para os algoritmos gerados para configurações de grupos de bases.

Configuração	F (AG)	F (AG-B)	F (AG-M)	F (AG-MB)	F (BG)	F (NB)	F (TAN)	F (K2)
1	0,652	0,656	0,655	0,659	0,655	0,655	0,609	0,649
7	0,872	0,824	0,830	0,825	0,866	0,836	0,823	0,830

pelo menos um dos cenários especificados.

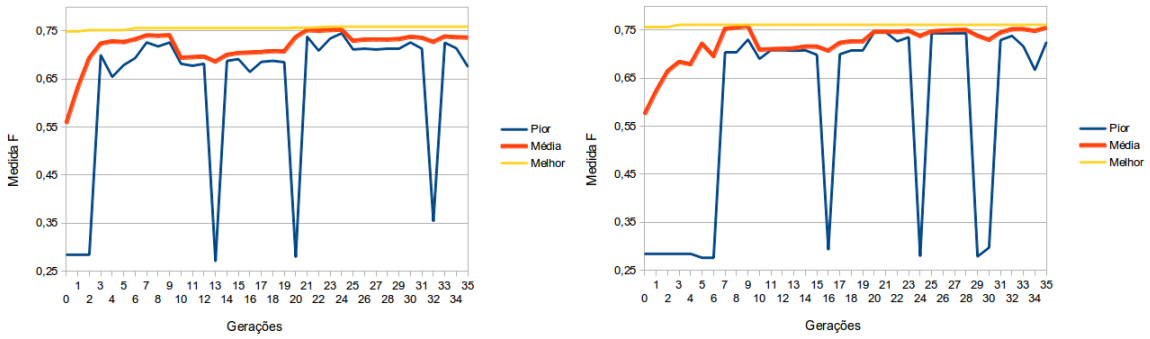
Os resultados mostraram as diferenças entre as soluções geradas pelas versões originais do genético (AG e AG-M) para os métodos de comparação. Para as versões inicializadas dos métodos, a busca conseguiu encontrar abordagens semelhantes aos métodos estado-da-arte, como visto na Tabela 4.31. A BG também se assemelhou aos métodos de comparação na Configuração 3. É importante frisar que esse comportamento das buscas ocorrem no restante das configurações (dos dados).

As curvas de evolução dos gráficos da Figura 4.3 (a-d) e 4.4 (a-d) mostram o acompanhamento do processo iterativo das versões do genético, em que o melhor indivíduo, a média de *fitness* da população e o pior indivíduo são monitorados a cada geração.



(a) Evolução do AG p/ Configuração 4.

(b) Evolução do AG-B p/ Configuração 4.

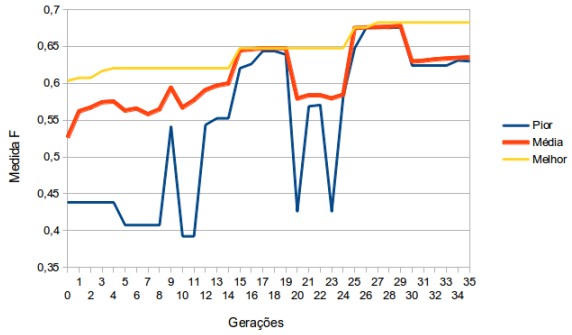


(c) Evolução do AG-M p/ Configuração 4.

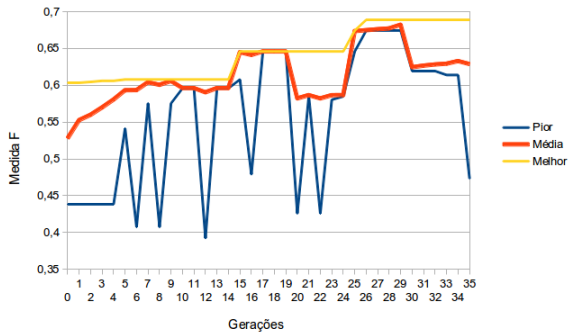
(d) Evolução do AG-MB p/ Configuração 4.

Figura 4.3: Curvas de evolução do AG, AG-B, AG-M e AG-MB para Configuração 4.

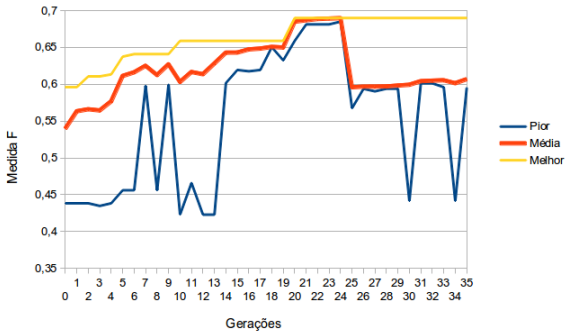
Analisa-se pelos gráficos da Figura 4.3 (a-d) e 4.4 (a-d) que quedas de *fitness* podem ocorrer geralmente na média da população e no pior indivíduo de forma mais acentuada. As explicações para as quedas bruscas de *fitness* e a reação quanto a



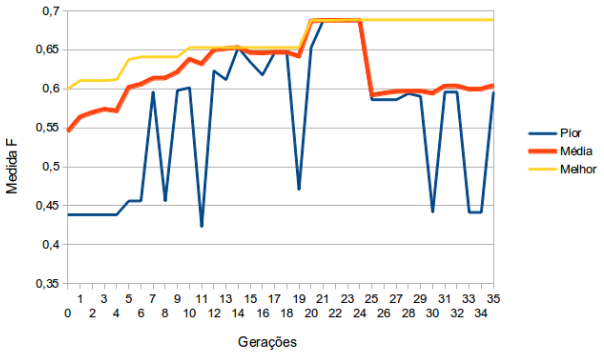
(a) Evolução do AG p/ Configuração 10.



(b) Evolução do AG-B p/ Configuração 10.



(c) Evolução do AG-M p/ Configuração 10.



(d) Evolução do AG-MB p/ Configuração 10.

Figura 4.4: Curvas de evolução do AG, AG-B, AG-M e AG-MB para Configuração 10.

inserção dos métodos de comparação na população inicial são similares aos da subseção anterior, a qual lida com conjunto de bases que estão presentes nos mesmos grupos.

Por fim, a Tabela 4.33 estima os tempos de execução das versões do algoritmo genético e da busca gulosa.

Tabela 4.33: Tempos médios de execução (segundos) para conjuntos de bases distintas.

Configuração	Cenário	AG	AG-B	AG-M	AG-MB	BG
1	1	336,000	4843,000	261,000	276,000	98,000
2	1	14197,000	5484,000	7336,000	13604,000	539,000
3	1	4506,000	2914,000	2877,000	5893,000	977,000
4	1	7768,000	5629,000	7866,000	2583,000	321,000
5	1	10127,000	2204,000	4573,000	3880,000	2068,000
6	1	1023,000	739,000	7916,000	2206,000	725,000
7	2	995,000	960,000	2151,000	1759,000	697,000
8	2	2951,000	788,000	4966,000	1558,000	529,000
9	2	303,000	684,000	426,000	1331,000	189,000
10	2	221,000	48,000	928,000	68,000	131,000

As medições de tempos de execução da Tabela 4.33 determinam que as Configurações 2, 5 e 10 demandam maior tempo computacional. Enquanto isso, as Configurações 1, 9 e 10 são executadas em menor tempo. Os tempos computacionais são influenciados principalmente pelas características das bases de dados de treinamento.

Capítulo 5

Conclusões e Trabalhos Futuros

Este trabalho propôs um método para evoluir automaticamente algoritmos de Redes Bayesianas de Classificação. O método é baseado em um algoritmo genético, onde cada indivíduo representa um algoritmo de RBC, construído a partir de um conjunto de componentes presentes na maioria dos algoritmos criados por projetistas humanos. A ideia principal deste trabalho é criar algoritmos de RBCs personalizados para uma ou mais bases de dados.

O foco desta pesquisa foi aplicado a bases de dados de domínio público da UCI, onde um conjunto de bases de características distintas foi selecionado para testes que almejam validar a técnica proposta.

Dada a natureza global do método de evolução de algoritmos proposto, dois tipos de comparações foram feitas: comparação do AG com um método local baseado em uma busca gulosa (BG) e comparação dos algoritmos gerados pelas versões do AG com três algoritmos estado-da-arte de RBCs: Naïve Bayes, *Tree Augmented* Naïve Bayes e K2.

Com os métodos de comparação definidos, os experimentos foram configurados e divididos em três etapas principais: (**i**) testes do método proposto em execuções direcionadas a bases de dados específicas, (**ii**) testes em execuções direcionadas a conjuntos de bases dados semelhantes e (**iii**) testes com conjuntos bases de treino e teste distintos.

Para a primeira etapa, o objetivo o algoritmo genético foi aprender um algoritmo personalizado para uma base de dados específica e gerar um modelo adequado para esses dados. Para essa etapa, 15 bases foram utilizadas e selecionadas nesses testes pela diferenças em suas características, como no tipo e no número de atributos e na quantidade de instâncias. Resultados mostraram que as versões da método proposto foram estatisticamente iguais aos resultados da BG e dos três métodos de comparação. Contudo, quando realizada uma análise pontual dos resultados gerados pelas versões

do AG, é possível concluir que as mesmas são capazes de superar o desempenho de classificação apresentado pelos métodos considerados como estado-da-arte.

Já para as duas demais etapas, concentrou-se na aplicação método proposto em conjuntos de bases de dados. Neste caso, 20 bases de dados foram selecionadas com o intuito de agrupá-las, podendo assim criar os diferentes cenários dos experimentos onde exista conjuntos de bases de treinamento e teste. Para o agrupamento, características das bases e métricas de complexidade de dados foram calculadas e utilizadas com esse propósito. Na segunda etapa, bases do mesmo grupo foram consideradas para treino e teste. E, para complementar a validação e verificar a importância de grupos de bases, a terceira etapa dos experimentos considera bases de diferentes grupos para treino e teste.

A segunda etapa possui a finalidade de verificar se os resultados do treinamento se mantêm, visto que a fase de teste engloba bases de dados com características de dados semelhantes segundo o agrupamento. Nessa etapa, observou-se que uma versão do algoritmo genético foi superior à busca gulosa e todas as versões do AG foram similares em relação a todos os métodos de comparação. Notou-se também que a inicialização do genético com os métodos de comparação melhorou os resultados, mas não a tal ponto de mostrar uma diferença estatística dos resultados da técnica proposta.

Por fim, na última etapa averiguou-se se o algoritmo generaliza, visto que o conjunto de bases de dados do treinamento pode distinguir bruscamente do conjunto de bases de dados do teste, pois as bases que constituem os conjuntos podem estar em grupos diferentes.

Os resultados para a última etapa mostram que as versões inicializadas do algoritmo genético foram superiores à busca gulosa e, além disso, as mesmas versões mantiveram-se equiparáveis aos métodos estado-da-arte utilizados para comparação. O mesmo não ocorre para as versões não-inicializadas, que se igualam a busca gulosa e possuem desempenho estatisticamente inferior aos métodos de comparação. Esse fato comprova que o uso da inicialização do AG é importante para encontrar boas soluções no espaço de busca de algoritmos de RBCs.

Dado os resultados encontrados, o primeiro trabalho futuro está em incluir uma busca local ao final do processo evolucionário, para que se possa considerar pontualmente todas as soluções geradas pelo AG. Essa nova busca feita em conjunto com o AG pode possivelmente melhorar os resultados gerados.

A mudança de como é feita a inicialização do AG também é uma alternativa interessante para outros trabalhos, visto que a inicialização trouxe melhorias dos resultados em relação às suas versões originais.

Pretende-se também considerar outras formas de representação que possam dis-

tinguir bem os componentes identificados. Dada a grande quantidade de dependências entre método de busca de estrutura e seus parâmetros, pode-se pensar ainda em uma forma de organização mais representativa dos componentes.

Adicionalmente, almeja-se que o trabalho proposto seja adaptado a bases de dados de outros contextos, como por exemplo aos de uma aplicação de transações online (identificação de transações fraudulentas) e/ou à problemas específicos da bioinformática (*Post-Synaptic Ativity in Proteins*, *Microarray Gene Expression* e *Flexible-Receptor Molecular Docking*). Trabalhos recentes mostraram que é em domínios reais que esses algoritmos se destacam [Barros et al., 2013a,b].

Referências Bibliográficas

Abramovici, M.; Neubach, M.; Fathi, M. & Holland, A. (2008). Competing fusion for bayesian applications. Em Magdalena, L.; Ojeda-Aciego, M. & Verdegay, J., editores, *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, volume 378-385.

Acid, S. & de Campos, L. M. (1996). BENEDICT: An algorithm for learning probabilistic belief networks. Em *Proceedings of the Sixth International Conference IPMU: Information Processing and Management of Uncertainty in Knowledge-based Systems*.

Acid, S. & de Campos, L. M. (2001). A hybrid methodology for learning belief networks: BENEDICT. *International Journal of Approximate Reasoning*, 27(3):235 – 262.

Asuncion, A. & Newman, D. (2007). UCI machine learning repository.

Back, T. & Schwefel, H.-P. (1996). Evolutionary computation: An overview. Em *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 20 – 29.

Bari, M. (2011). Bayesian network structure learning. Em *Proceeding of the 4th Annual Meeting of Asian Association for Algorithms and Computation (AAAC), April 16-17, HsinChu, Taiwan.*

Barros, R.; Basgalupp, M.; Freitas, A. & de Carvalho, A. (2013a). Evolutionary design of decision-tree algorithms tailored to microarray gene expression data sets. *Evolutionary Computation, IEEE Transactions on*, PP(99):1–1.

Barros, R. C.; Basgalupp, M. P.; de Carvalho, A. C. & Freitas, A. A. (2012). A hyper-heuristic evolutionary algorithm for automatically designing decision-tree algorithms. Em *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pp. 1237 – 1244, New York, NY, USA. ACM.

Barros, R. C.; Basgalupp, M. P.; de Carvalho, A. C. P. L. F. & Freitas, A. A. (2013b). Automatic design of decision-tree algorithms with evolutionary algorithms. *Evolutionary Computation (MIT)*, 21(4):659–684.

Beinlich, I. A.; Suermondt, H. J.; Chavez, R. M. & Cooper, G. F. (1989). The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. Em Hunter, J.; Cookson, J. & Wyatt, J., editores, *Second European Conference on Artificial Intelligence in Medicine*, volume 38, pp. 247 – 256, Berlin, Germany. Springer-Verlag.

Bouckaert, R. (1995). *Bayesian Belief Networks: from Construction to Inference*. Tese de doutorado, Utrecht, Netherlands.

Bouckaert, R. R.; Frank, E.; Hall, M.; Kirkby, R.; Reutemann, P.; Seewald, A. & Scuse, D. (2013). *WEKA Manual for Version 3-6-9*. University of Waikato, Hamilton, New Zealand.

Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(2):123 – 140.

Cantu-Paz, E. & Kamath, C. (2005). An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(5):915 – 927.

Cheng, J.; Bell, D. A. & Liu, W. (1997a). An algorithm for Bayesian belief network construction from data. Em *In Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 83 – 90.

Cheng, J.; Bell, D. A. & Liu, W. (1997b). Learning belief networks from data: an information theory based approach. Em *Proceedings of the sixth international conference on Information and knowledge management*, CIKM '97, pp. 325 – 331, New York, NY, USA. ACM.

Cheng, J.; Bell, D. A. & Liu, W. (1998). Learning Bayesian networks from data: An efficient approach based on information theory. Relatório técnico, University of Alberta.

Cheng, J. & Greiner, R. (1999). Comparing Bayesian network classifiers. Em *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, UAI'99, pp. 101 – 108, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Cheng, J. & Greiner, R. (2001). Learning Bayesian belief network classifiers: Algorithms and system. Em *Proceedings of the 14th Biennial Conference of the Canadian*

Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, AI '01, pp. 141 – 151, London, UK, UK. Springer-Verlag.

Cheng, J.; Greiner, R.; Kelly, J.; Bell, D. & Liu, W. (2002). Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1 - 2):43 – 90.

Chow, C. & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462 – 467.

Cooper, G. F. & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309 – 347.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, 3rd edição.

Daly, R.; Shen, Q. & Aitken, S. (2011). Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, 26(02):99 – 157.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1--30.

Eiben, A. E. & Smith, J. E. (2003). *Introduction to Evolutionary Computing*. SpringerVerlag.

Floreano, D.; Durr, P. & Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1:47 – 62.

Freund, Y. & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. Em *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pp. 23 – 37, London, UK, UK. Springer-Verlag.

Friedman, M. (1937). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675--701.

Friedman, N.; Geiger, D. & Goldszmidt, M. (1997). Bayesian network classifiers. *Mach. Learn.*, 29(2 - 3):131 – 163.

Giraud-Carrier, C.; Vilalta, R. & Brazdil, P. (2004). Introduction to the special issue on meta-learning. *Mach. Learn.*, 54(3):187 – 193.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P. & Witten, I. H. (2009). The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10 – 18.

Heckerman, D. (1997). Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1):79 – 119.

Herskovits, E. & Cooper, G. F. (1991). Kulató: An entropy-driven system for construction of probabilistic expert systems from databases. Em *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, pp. 117 – 128, New York, NY, USA. Elsevier Science Inc.

Hesar, A. S.; Tabatabaee, H. & Jalali, M. (2012). Structure learning of bayesian networks using heuristic methods. Em *Proceedings of International Conference on Information and Knowledge Management (ICIKM 2012)*.

Ho, T.; Basu, M. & Law, M. (2006). Measures of geometrical complexity in classification problems. Em Basu, M. & Ho, T., editores, *Data Complexity in Pattern Recognition*, Advanced Information and Knowledge Processing, pp. 1–23. Springer London.

Ho, T. K. & Basu, M. (2002). Complexity measures of supervised classification problems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):289–300.

Iman, R. & Davenport, J. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics*, pp. 571–595.

Krieg, M. L. (2001). A tutorial on Bayesian belief networks. Relatório técnico, Surveillance Systems Division, DSTO Electronics and Surveillance Research Laboratory.

Kullback, S. & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79--86.

Lam, W. & Bacchus, F. (1994). Learning bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, 10:269--293.

Land, A. H. & Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497--520.

Langley, P.; Iba, and, W. & Thompson, K. (1992). An analysis of Bayesian classifiers. Em *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, pp. 223 – 228. AAAI Press.

Lawler, E. L. & Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719.

Orriols-Puig, A.; MaciÃ , N. & Ho, T. K. (2010). Dcol: Data complexity library in c++ (documentation). Relatório técnico, La Salle - Universitat Ramon Llull.

Pappa, G. L. & Freitas, A. A. (2009a). Automatically evolving rule induction algorithms tailored to the prediction of postsynaptic activity in proteins. *Intelligent Data Analysis*, 13(2):243 – 259.

Pappa, G. L. & Freitas, A. A. (2009b). *Automating the Design of Data Mining Algorithms: An Evolutionary Computation Approach*. Springer.

Pappa, G. L. & Freitas, A. A. (2009c). Evolving rule induction algorithms with multi-objective grammar-based genetic programming. *Knowl. Inf. Syst.*, 19(3):283 – 309.

Pelleg, D. & Moore, A. W. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. Em *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 727--734, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Peng, H. & Ding, C. (2003). Structure search and stability enhancement of bayesian networks. Em *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, pp. 621 – 624, Los Alamitos, CA, USA. IEEE Computer Society.

Raidl, G. R. (2005). Evolutionary computation: An overview and recent trends. *OGAI Journal*, 24:2 – 7.

Rebane, G. & Pearl, J. (1987). The recovery of causal poly-trees from statistical data. Em *Uncertainty in Artificial Intelligence 3 Annual Conference on Uncertainty in Artificial Intelligence (UAI-87)*, pp. 175 – 182, Amsterdam, NL. Elsevier Science.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465 – 471.

Sá, A. G. C. & Pappa, G. L. (2013). Towards a method for automatically evolving bayesian network classifiers. Em *Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion*, GECCO '13 Companion, pp. 1505--1512, New York, NY, USA. ACM.

Sacha, J. (1999a). Java Bayesian network classifier toolbox (jbnc toolkit). Disponível em: *http://jbnc.sourceforge.net/*.

Sacha, J. P. (1999b). *New synthesis of bayesian network classifiers and cardiac spect image interpretation*. Tese de doutorado.

Sacha, J. P.; Goodenday, L. S. & Cios, K. J. (2002). Bayesian learning for cardiac spect image interpretation. *Artif. Intell. Med.*, 26(1 - 2):109 – 143.

Salama, K. & Freitas, A. (2014). Extending the abc-miner bayesian classification algorithm. Em Terrazas, G.; Otero, F. E. B. & Masegosa, A. D., editores, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, volume 512 of *Studies in Computational Intelligence*, pp. 1–12. Springer International Publishing.

Santos, E. B.; Hruschka, Jr., E. R.; Hruschka, E. R. & Ebecken, N. F. F. (2011). Bayesian network classifiers: Beyond classification accuracy. *Intell. Data Anal.*, 15(3):279 – 298.

Singh, M. & Valtorta, M. (1995). Construction of bayesian network structures from data: a brief survey and an efficient algorithm. pp. 259 – 265.

Spirtes, P. & Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62 – 72.

Srinivas, S.; Russell, S. J. & Agogino, A. M. (1990). Automated construction of sparse bayesian networks from unstructured probabilistic models and domain information. Em *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '89, pp. 295 – 308, Amsterdam, The Netherlands, The Netherlands. North-Holland Publishing Co.

Stanley, K. O. & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10:99 – 127.

Suzuki, J. (1999). Learning bayesian belief networks based on the mdl principle: An efficient algorithm using the branch and bound technique. *IEICE Transactions on Information and Systems*, E82-D:356 – 367. Institute of Electronics, Information and Communication Engineers.

Verma, T. & Pearl, J. (1992). An algorithm for deciding if a set of observed independencies has a causal explanation. Em *Proc. of the Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 323--330. Morgan Kaufmann.

Vilalta, R. & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artif. Intell. Rev.*, 18(2):77 – 95.

Witten, I. H.; Frank, E. & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edição.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5:241 – 259.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67--82.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423 – 1447.