DEBORAH RIBEIRO CARVALHO

UM MÉTODO HÍBRIDO ÁRVORE DE DECISÃO / ALGORITMO GENÉTICO PARA DATA MINING

Tese apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutor em Informática Aplicada.

Área de Concentração: *Sistemas Inteligentes*

Orientador: Prof. Dr Alex Alves Freitas

CURITIBA

2002

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática Aplicada

ATA DA SESSÃO PÚBLICA DE DEFESA DE TESE DE DOUTORADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

DEFESA DE TESE Nº 004

Aos 17 dias do mês de junho de 2002 realizou-se a sessão pública de defesa da tese "**Um Método Híbrido Árvore de Decisão / Algoritmos Genéticos para Data Mining**", apresentada por Deborah Ribeiro Carvalho como requisito parcial para a obtenção do título de **Doutor em Informática Aplicada**, perante uma Banca Examinadora composta pelos seguintes membros:

	assinatura	parecer (aprov/reprov)
Prof. Dr. Alex Alves de Freitas PUCPR (Presidente)	Alex Alves Freitas	APROVADO
Prof. Dr. Celso A. A. Kaestner PUCPR		APROVADO
Prof. Dr. Júlio Cesar Nievola PUCPR	Júlio Cesar Nievola	APROVADO
Prof. Dr. Heitor S. Lopes CEFET-PR	Heitor S. Lopes	APROVADO
Prof. Dr. Nelson F. F. Ebecken UFRJ	Nelson Francisco Favilla Ebecken	APROVADO

Conforme as normas regimentais do PPGIA e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora, conforme registrado no Livro de Defesas do programa.

Prof. Dr. Carlos Maziero
Diretor do PPGIA PUCPR 02/08/02 Carlos Maziero
Data e assinatura, após homologação da defesa pelo colegiado.

AGRADECIMENTOS

Agradeço ao Prof. Dr. Ale A. Freitas pela orientação acadêmica e o constante apoio durante todo o período de doutoramento e de desenvolvimento do trabalho. Aos demais professores do Programa de pós-graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná, em especial ao Prof. Dr. Celso A. A. Kaestner e ao Prof. Dr. Júlio C. Nievola, pelas aulas ministradas, discussões e esclarecimento de dúvidas sobre este trabalho. Ao Prof. Dr. Heitor S. Lopes como professor da disciplina Computação Evolucionária (CEFET-PR) e o apoio técnico-científico na fase embrionária do trabalho.

Agradeço à Pontifícia Universidade Católica pela oportunidade de realizar este curso e pela concessão de uma bolsa de estudos constituindo uma redução significativa no valor da mensalidade.

Agradeço à Universidade Tuiuti do Paraná pela permissão de uso de seus laboratórios para a execução de grande parte dos experimentos relatados no trabalho. Agradeço aos professores da Universidade Tuiuti do Paraná pelo constante apoio durante o período do curso, trocas de horário de aulas, idéias sobre algumas questões conceituais ou mesmo de apresentação.

Agradeço aos meus colegas que trabalham no IPARDES – Instituto Paranaense de Desenvolvimento Econômico e Social pelo apoio durante o período de realização deste curso.

Agradeço ao Prof. Dr. Wesley Romão da Universidade Estadual de Maringá pela disponibilização de algumas bases de dados que foram utilizadas nos experimentos.

E, finalmente, agradeço às pessoas da minha família, aos meus pais, ao meu marido e aos meus quatro filhos, pois estou convencida de que somente as pessoas que dispõem de tal apoio concluem uma atividade com nível de dedicação tão elevado.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho aborda a tarefa de classificação em *Data Mining*, onde o conhecimento pode ser representado por regras da forma "se-então". Em síntese, pequenos disjuntos são regras que cobrem um pequeno número de exemplos. Apesar de cada pequeno disjunto cobrir um pequeno número de exemplos, o conjunto de todos os pequenos disjuntos pode coletivamente cobrir um grande número de exemplos. Portanto, é importante descobrir regras que cubram efetivamente exemplos de pequenos disjuntos. Este trabalho propõe um método híbrido árvore de decisão / algoritmo genético para tratar do problema dos pequenos disjuntos. A idéia básica é que os exemplos pertencentes a grandes disjuntos sejam classificados pelas regras produzidas pelo algoritmo de árvore de decisão, enquanto exemplos pertencentes a pequenos disjuntos (aqueles considerados de mais difícil classificação) sejam classificados pelas regras descobertas por um algoritmo genético. Além disso, este trabalho propõe dois algoritmos genéticos especificamente projetados para a descoberta de regras de pequenos disjuntos. O método híbrido proposto é avaliado comparando-o com vários outros algoritmos em 22 bases de dados. O método proposto obteve bons resultados, considerando tanto a precisão preditiva quanto a simplicidade das regras descobertas.

Palavras-chave: *Data Mining*, Classificação, Algoritmo Genético, Árvore de Decisão, Pequenos Disjuntos

RESUMO

Este trabalho aborda a tarefa de classificação em *Data Mining*, onde o conhecimento pode ser representado por regras da forma "se-então". Em síntese, pequenos disjuntos são regras que cobrem um pequeno número de exemplos. Apesar de cada pequeno disjunto cobrir um pequeno número de exemplos, o conjunto de todos os pequenos disjuntos pode coletivamente cobrir um grande número de exemplos. Portanto, é importante descobrir regras que cubram efetivamente exemplos de pequenos disjuntos. Este trabalho propõe um método híbrido árvore de decisão / algoritmo genético para tratar do problema dos pequenos disjuntos. A idéia básica é que os exemplos pertencentes a grandes disjuntos sejam classificados pelas regras produzidas pelo algoritmo de árvore de decisão, enquanto exemplos pertencentes a pequenos disjuntos (aqueles considerados de mais difícil classificação) sejam classificados pelas regras descobertas por um algoritmo genético. Além disso, este trabalho propõe dois algoritmos genéticos especificamente projetados para a descoberta de regras de pequenos disjuntos. O método híbrido proposto é avaliado comparando-o com vários outros algoritmos em 22 bases de dados. O método proposto obteve bons resultados, considerando tanto a precisão preditiva quanto a simplicidade das regras descobertas.

Palavras-chave: *Data Mining*, Classificação, Algoritmo Genético, Árvore de Decisão, Pequenos Disjuntos

ABSTRACT

This work addresses the classification task in Data Mining, where the discovered knowledge can be represented by "if-then" rules. In essence, small disjuncts are rules covering a small number of examples. Although each small disjunct covers a small number of examples, the set of all small disjuncts can collectively cover a large number of examples. Therefore, it is important to discover rules that effectively cover small-disjunct examples. This work proposes a hybrid decision tree / genetic algorithm method for coping with the small-disjunct problem. The basic idea is that examples belonging to large disjuncts are classified by a decision tree algorithm, whereas examples belonging to small disjuncts (whose classification is considerably more difficult) are classified by rules discovered by a genetic algorithm. In addition, this work proposes two genetic algorithms specifically designed for discovering small disjunct rules. The proposed hybrid method is evaluated by comparing it with several other algorithms in 22 data sets. The proposed method obtained good results, with respect to both the predictive accuracy and the simplicity of the discovered rules.

Keywords: Data Mining, Classification, Genetic Algorithm, Decision Tree, Small Disjuncts

ABSTRACT

This work addresses the classification task in Data Mining, where the discovered knowledge can be represented by "if-then" rules. In essence, small disjuncts are rules covering a small number of examples. Although each small disjunct covers a small number of examples, the set of all small disjuncts can collectively cover a large number of examples. Therefore, it is important to discover rules that effectively cover small-disjunct examples. This work proposes a hybrid decision tree / genetic algorithm method for coping with the small-disjunct problem. The basic idea is that examples belonging to large disjuncts are classified by a decision tree algorithm, whereas examples belonging to small disjuncts (whose classification is considerably more difficult) are classified by rules discovered by a genetic algorithm. In addition, this work proposes two genetic algorithms specifically designed for discovering small disjunct rules. The proposed hybrid method is evaluated by comparing it with several other algorithms in 22 data sets. The proposed method obtained good results, with respect to both the predictive accuracy and the simplicity of the discovered rules.

Keywords: Data Mining, Classification, Genetic Algorithm, Decision Tree, Small Disjuncts

1 INTRODUÇÃO

O ser humano continuamente toma decisões ou simplesmente chega a determinadas conclusões baseadas no conhecimento que ele acumula ao longo de sua vida.

Aliado ao fato do conhecimento fazer parte de nosso dia-a-dia, vários motivos contribuíram para que a aquisição de conhecimento se tornasse objeto de pesquisa e investimento na área da computação.

Dentre estes motivos, pode-se citar os seguintes [DEC95]:

- a disponibilidade de grande capacidade de processamento a baixo custo;
- a geração de grande volume de dados em função do desenvolvimento científico e tecnológico, tornando-os impossíveis de serem analisados pelos métodos tradicionais de armazenamento e de recuperação de dados;
- o surgimento de um novo conjunto de métodos para análise de dados desenvolvidos principalmente pelas comunidades de Inteligência Artificial e de Estatística[1].

A atual "era da informação" é caracterizada pela grande expansão no volume de dados gerados e armazenados [HAN99]. Uma grande parte destes dados estão armazenados em bases de dados e podem ser facilmente acessáveis pelos seus usuários.

Esta situação tem gerado demandas por novas técnicas e ferramentas que, com eficiência, transformem os dados armazenados e processados em conhecimento. Este é o motivo pelo qual *Data Mining*/descoberta de conhecimento é uma área de pesquisa com grande interesse [MIC98].

Nesse sentido, algumas das tarefas de *Data Mining* mais populares são a classificação e o *clustering* [FAY96].

No desenvolvimento desta tese, o foco é na classificação, onde o objetivo é atribuir uma classe a um exemplo (registro), dentre um conjunto de classes pré-definidas, com base nos valores de seus atributos.

---

[1] Historicamente, o desenvolvimento da Estatística tem resultado em um grande número de métodos de análise, que têm sido amplamente aplicados com o objetivo de encontrar correlações e dependências nos conjuntos de dados.

Cabe ressaltar que essa tarefa tem várias aplicações em diversas áreas. Por exemplo, em uma aplicação financeira um banco poderia classificar seus clientes em duas classes: "crédito ruim" ou "crédito bom". Em uma aplicação de medicina, um médico poderia classificar alguns de seus pacientes em duas classes: "tem" ou "não tem" uma certa doença.

De forma semelhante, vários problemas importantes do mundo real podem ser modelados através da classificação. Essa tarefa é discutida de forma mais detalhada na seção 2.1.

1.1 DEFINIÇÃO DO PROBLEMA E OBJETIVO

No contexto da tarefa de classificação, o conhecimento descoberto pode ser expresso através de um conjunto de regras "se ... então". Este tipo de representação tem a vantagem de ser intuitivamente compreensível para o usuário. Do ponto de vista da representação em lógica, as regras descobertas geralmente estão na forma normal disjuntiva, onde cada regra representa um disjunto.

Nesse contexto, um pequeno disjunto pode ser definido como uma regra que cobre um pequeno número de exemplos de treinamento [HOL89]. Mais precisamente, nesta tese, adotou-se a seguinte definição de pequeno disjunto: uma regra é considerada um pequeno disjunto se e somente se o número de exemplos "cobertos" pela regra é menor ou igual a um *threshold S*, definido pelo usuário. Um exemplo é dito "coberto" por uma regra se este satisfaz todas as condições especificadas no antecedente da regra. A justificativa para essa definição de pequenos disjuntos será discutida na seção 4.3.

Em geral algoritmos de indução de regras têm um *bias* que favorece a descoberta de grandes disjuntos, ao invés de pequenos disjuntos. Isto se deve à crença de que especializações no conjunto de treinamento são indesejáveis na validação sobre o conjunto de teste ou ao argumento de que os pequenos disjuntos não deveriam ser incluídos no conjunto de regras descobertas, uma vez que eles tendem a ser uma das causas de erros na classificação dos dados de teste.

Entretanto, apesar de cada pequeno disjunto cobrir um pequeno número de exemplos, o conjunto de todos os pequenos disjuntos pode cobrir um grande número de exemplos. Por exemplo, Danyluk e Provost [DAN93] relatam que, em uma aplicação do mundo real, o conjunto dos pequenos disjuntos pode vir a cobrir em torno de 50% dos exemplos de treinamento.

Desta forma, se o algoritmo de indução de regras ignora os pequenos disjuntos e descobre apenas regras que cubram grandes disjuntos, a precisão preditiva do processo de classificação pode ser reduzida de forma significativa. Assim, pequenos disjuntos são um problema importante em aprendizado de máquina e *Data Mining* [WEI00].

Esta tese tem como objetivo principal propor um novo método para tratar o problema do pequeno disjunto. A idéia básica desse método é que os exemplos pertencentes a grandes disjuntos devem ser classificados por regras produzidas por um algoritmo de árvore de decisão, enquanto os exemplos pertencendo a pequenos disjuntos devem ser classificados por regras produzidas por algoritmos evolucionários especificamente projetados para descobrir regras de pequenos disjuntos.

A contribuição desta tese está relacionada com as questões de originalidade do método proposto e relevância dos resultados computacionais, conforme serão discutidas a seguir.

Dadas as atividades de pesquisa bibliográfica realizada no âmbito dessa tese, há indicações de que o problema de pequenos disjuntos é relativamente pouco investigado na literatura. Conforme será visto no Capítulo 5 que discute trabalhos relacionados ao tema, foram encontrados poucos exemplos com foco nesse problema. Além disso, a maioria dos trabalhos relacionados se concentra em discutir a importância do problema, sem propor novas soluções.

Neste contexto, o novo método proposto nesta tese parece ser uma importante contribuição para solução do problema de pequenos disjuntos. Além disso, um grande número de experimentos computacionais foi realizado com o objetivo de avaliar o método proposto, comparando-o com vários outros algoritmos (árvore de decisão e algoritmo genético classificando tanto grandes como pequenos disjuntos e um híbrido composto por árvore de decisão e algoritmo de aprendizado baseado em instâncias).

No geral, os resultados obtidos foram bons, tanto em relação à precisão preditiva quanto à simplicidade das regras descobertas, conforme será mostrado no Capítulo 4.

1.2 ORGANIZAÇÃO DO TRABALHO

Esta tese, além da Introdução, inclui os seguintes capítulos.

No Capítulo 2 são apresentadas e discutidas as principais características de *Data Mining*, suas tarefas e algoritmos, com uma maior ênfase à tarefa de classificação e aos dois

tipos de algoritmos (árvore de decisão e algoritmo genético) que compõem o método híbrido proposto nesta tese.

No Capítulo 3 este método é apresentado e analisado o seu potencial para resolver o problema de pequenos disjuntos.

No Capítulo 4 são apresentados resultados computacionais alcançados pelo método híbrido, bem como a comparação desses resultados com os obtidos a partir dos outros algoritmos utilizados.

No Capítulo 5 são descritos alguns trabalhos relacionados ao tema dos pequenos disjuntos.

Finalmente, o Capítulo 6 sumariza os resultados, apresenta as conclusões e indica algumas direções de trabalhos futuros.

2 *DATA MINING*

Data mining consiste em um conjunto de conceitos e métodos com o objetivo de encontrar uma descrição, preferencialmente compreensível, de padrões e regularidades em um determinado conjunto de dados.

Os termos *Data mining* e Descoberta de Conhecimento em Base de *Dados (Knowledge Discovery in Databases* – KDD) muitas vezes são confundidos como sinônimos para identificar o processo de descoberta de informação útil de bancos de dados. O termo KDD foi estabelecido no primeiro *workshop* de KDD em 1989 para enfatizar que conhecimento é o produto final de uma descoberta baseada em dados (*data-driven*). Desta forma KDD se refere a todo o processo de descoberta de conhecimento enquanto *Data Mining* se refere a uma das etapas deste processo. As etapas do KDD envolvem preparação dos dados, seleção, limpeza, transformação, *Data Mining* e interpretação dos resultados [FAY98].

Um padrão é definido como um tipo de declaração (ou modelo de uma declaração) sobre o conjunto de dados que está sendo analisado. Uma instância de um padrão é uma declaração em uma linguagem de alto nível que descreve uma informação interessante descoberta nos dados. A descoberta de relações nos dados compreende todas as instâncias de padrões selecionados no espaço das hipóteses que sejam suficientemente interessantes, de acordo com algum critério estabelecido [KLO92].

As várias tarefas desenvolvidas em *Data Mining* têm como objetivo primário a predição e/ou a descrição. A predição usa atributos para predizer os valores futuros de uma ou mais variáveis (atributos) de interesse. A descrição contempla o que foi descoberto nos dados sob o ponto de vista da interpretação humana [FAY96].

2.1 TAREFAS DE *DATA MINING*

O objetivo da descrição, bem como o da predição, são atendidos através de algumas das tarefas principais de *Data Mining* (classificação, *clustering*, etc.) [ADR96], [FAY96], [FU96]. A seguir são descritas duas dessas tarefas: a da classificação, que é o foco deste trabalho, e a de *clustering*, a qual pode ser utilizada para análise inicial dos dados, possivelmente levando posteriormente à execução da tarefa de classificação, conforme será explicado adiante.

Classificação

A classificação, por vezes chamada de aprendizado supervisionado [FIS93], parece ser a tarefa de *Data Mining* que tem sido mais estudada ao longo do tempo. Essa tarefa consiste em classificar um item de dado (exemplo ou registro) como pertencente a uma determinada classe dentre várias classes previamente definidas.

Cada classe corresponde a um padrão único de valores dos atributos previsores (demais atributos que caracterizam o exemplo). Esse padrão único pode ser considerado a descrição da classe. O conjunto de todas as classes é definido como C, e a cada classe C_i, correspondente uma descrição D_i das propriedades selecionadas. Desta forma, usando estas descrições é possível construir um classificador o qual descreve um exemplo e do conjunto de exemplos T como sendo um exemplo pertencendo à classe C_i, quando aquele exemplo satisfaz D_i.

O principal objetivo da construção de um classificador é descobrir algum tipo de relação entre os atributos e as classes [FRE98]. Por exemplo, na Figura 2.1 o classificador em questão tem como objetivo a identificação da relação existente entre os atributos previsores A_1 e A_2 e os valores da classe ("+" e "-"). O procedimento de construção deste classificador é baseado em particionamentos recursivos do espaço de dados. O espaço é dividido em áreas e a cada estágio é avaliado se cada área deve ser dividida em subáreas, a fim de obter uma separação das classes.

Segundo Breiman e seus colegas [BRE84] um classificador extraído de um conjunto de dados serve a dois propósitos: predição de um valor e entender a relação existente entre os atributos previsores e a classe. Para cumprir o segundo propósito é exigido do classificador que ele não apenas classifique, mas também explicite o conhecimento extraído da base de dados de forma compreensível.

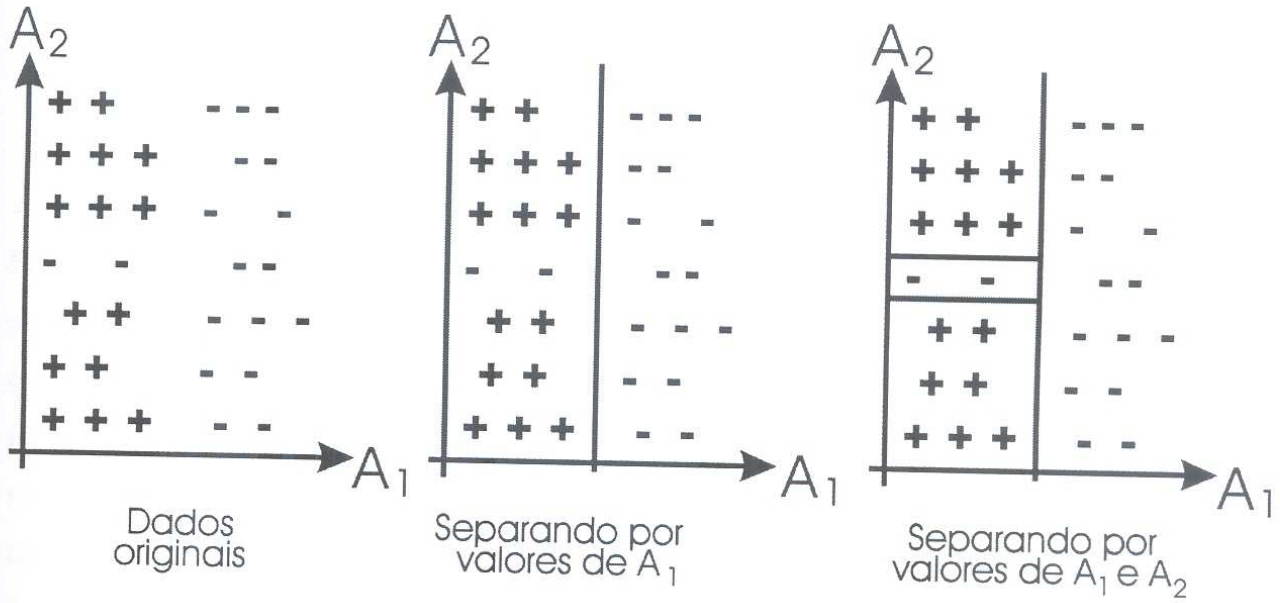


FIGURA 2.1 EXEMPLO DE CLASSIFICAÇÃO [FRE98]

A fim de contribuir para a compreensibilidade do conhecimento descoberto (relação entre os atributos e as classes), esse conhecimento é geralmente representado na forma de regras "se"..(condições).. "então"..(classe).., cuja interpretação é: "se" os valores dos atributos satisfazem as condições da regra "então" o exemplo pertence à classe prevista pela regra.

Essa forma de representação de conhecimento será adotada neste trabalho, em virtude de ser intuitivamente compreensível para o usuário.

Existem vários critérios para avaliar a qualidade das regras descobertas na tarefa de classificação. Os dois critérios mais usados são precisão preditiva e a compreensibilidade do conhecimento descoberto.

Precisão preditiva é normalmente medida como o número de exemplos de teste classificados corretamente dividido pelo número total de exemplos de teste. Cabe ressaltar que há formas mais sofisticadas de se medir a precisão preditiva [HAN97], mas a forma simples descrita anteriormente é, em sua essência, a forma mais usada na prática. A compreensibilidade geralmente é medida pela simplicidade, qual por sua vez é medida em função do número de regras e do número de condições por regra. Quanto maiores estes números, menos compreensível é o conjunto de regras em questão. É importante ressaltar que mesmo nos paradigmas que tradicionalmente têm a característica de descobrir conhecimento expresso sob uma forma dita compreensível, algumas vezes pode ser gerado um modelo muito complexo, o qual dificilmente satisfaz o requisito de compreensibilidade. Este fato pode decorrer da complexidade existente entre os atributos previsores e as classes, o nível de ruído existente nos dados, etc. [FRE98].

Um importante conceito da tarefa de classificação é a divisão dos dados entre dados de treinamento e dados de teste. Inicialmente, um conjunto de dados de treinamento é disponibilizado e analisado, e um modelo de classificação é construído baseado nesses dados. Então o modelo construído é utilizado para classificar outros dados, chamados dados de teste, os quais não foram contemplados pelo algoritmo durante a fase de treinamento. Cabe ressaltar que o modelo construído a partir dos dados de treinamento só será considerado um bom modelo, do ponto de vista de precisão preditiva, se ele classificar corretamente uma alta porcentagem dos exemplos (registros) dos dados de teste. Em outras palavras, o modelo deve representar conhecimento que possa ser generalizado para dados de teste, não utilizados durante o treinamento.

Clustering

A tarefa de *clustering*, às vezes chamada de classificação não-supervisionada [CHE96], consiste na identificação de um conjunto finito de classes ou *clusters*, baseada nos atributos dos objetos não previamente classificados. Um *cluster* é basicamente um conjunto de objetos agrupados em função de sua similaridade ou proximidade. Os objetos são agrupados de tal forma que as similaridades intraclusters (dentro de um mesmo *cluster*) sejam maximizadas e as similaridades interclusters (entre *clusters* diferentes) sejam minimizadas. A Figura 2.2 mostra um exemplo do resultado de uma tarefa de *clustering*, onde quatro *clusters* foram identificados.

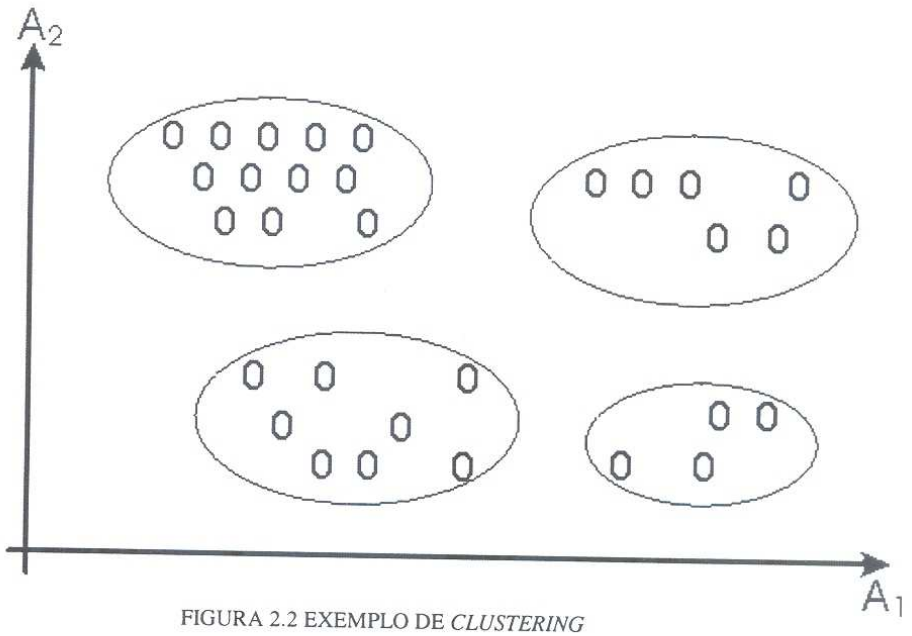


FIGURA 2.2 EXEMPLO DE *CLUSTERING*

Uma vez definidos os *clusters*, os objetos são identificados com seu *cluster* correspondente, e as características comuns dos objetos no *cluster* podem ser sumarizadas para formar a descrição da classe. Por exemplo, um conjunto de pacientes pode ser agrupado em várias classes (*clusters*) baseado nas similaridades dos seus sintomas, e os sintomas comuns aos pacientes de cada *cluster* podem ser usados para descrever à qual *cluster* um novo paciente pertencerá. Assim, um dado paciente seria atribuído ao *cluster* cujos pacientes têm sintomas o mais parecido possível com os sintomas daquele dado paciente. Dessa forma, a tarefa de *clustering*, cujo resultado é a identificação de novas classes, pode ser realizada como pré-processamento para realização da tarefa de classificação [KUB98].

2.2 ÁRVORE DE DECISÃO

Uma árvore de decisão é uma representação simples de um classificador utilizada por diversos sistemas de aprendizado de máquina, como por exemplo o C4.5 [QUI93].

Uma árvore de decisão é induzida a partir de um conjunto de exemplos de treinamento onde as classes são previamente conhecidas. A estrutura da árvore é organizada de tal forma que:

(a) cada nó interno (não-folha) é rotulado com o nome de um dos atributos previsores;

(b) os ramos (ou arestas) saindo de um nó interno são rotulados com valores do atributo naquele nó;

(c) cada folha é rotulada com uma classe, a qual é a classe prevista para exemplos que pertençam àquele nó folha.

O processo de classificação de um exemplo ocorre fazendo aquele exemplo "caminhar" pela árvore, a partir do nó raiz, procurando percorrer os arcos que unem os nós, de acordo com as condições que estes mesmos arcos representam. Ao atingir um nó folha, a classe que rotula aquele nó folha é atribuída àquele exemplo.

No espaço definido pelos atributos, cada nó folha corresponde a uma região, um hiper-retângulo, onde a interseção dos hiper-retângulos é o conjunto vazio e a união destes hiper-retângulos é o espaço completo. Sob este ponto de vista, um disjunto nada mais é do que um hiper-retângulo.

2.2.1 Algoritmo para Indução de Árvores de Decisão

Um algoritmo para indução de árvores de decisão trata-se de um exemplo de algoritmo de estrutura TDIDT - *Top-Down Induction of Decision Trees*. Este algoritmo utiliza a estratégia "dividir-para-conquistar", ou seja, um problema complexo é decomposto em subproblemas mais simples.

Esse algoritmo consiste dos procedimentos descritos na Figura 2.3, os quais criam uma árvore que classifica todos os exemplos do conjunto de treinamento corretamente. A idéia básica do algoritmo (Figura 2.3) é:

a) escolher um atributo;
b) estender a árvore adicionando um ramo para cada valor do atributo;
c) passar os exemplos para as folhas (tendo em conta o valor do atributo escolhido);
d) para cada nó folha – se todos os exemplos são da mesma classe, associar esta classe ao nó folha, caso contrário, repetir os passos (a), (b) e (c).

Porém, a árvore assim construída pode estar ajustada demais (*overfitted*) aos dados de treinamento. Uma árvore de decisão *a* está ajustada demais aos dados se existir uma árvore *a'* tal que *a* tem menor erro que *a'* no conjunto de treinamento, porém *a'* tem menor erro no conjunto de teste.

Para corrigir o fato de uma árvore estar ajustada demais, deve-se executar um procedimento de poda da árvore, como será explicado posteriormente. Antes disso, porém, serão apresentados os principais conceitos usados na construção da árvore.

```
/* Conj_Exemplos representa o conjunto de treinamento */
/* Atributo_Meta é o atributo a ser predito pela árvore */
/* Lista_Atributos representa a lista dos outros atributos a serem testados*/

INICIO1 (Conj_Exemplos , Atributo_Meta , Lista_Atributos)
Selecionar o melhor atributo para o nó raiz da árvore, de acordo com função de avaliação
   SE todos os exemplos em Conj_Exemplos são de uma única classe
      ENTÃO
         Retornar um único nó com valor da classe
      CASOCONTRÁRIO
          SE Lista_Atributos = ϕ
              ENTÃO
                  Retornar um único nó com o valor de Atributo_Meta mais freqüente em Conj_Exemplos
              CASOCONTRÁRIO
                  INICIO2
                     A ← o atributo de Lista_Atributos que melhor classifica Conj_Exemplos
                     PARA cada valor (v_i) possível de A
                        Adicionar uma nova ramificação, A = v_i
                        Criar o subconjunto Conj_Exemplos_vi contendo os exemplos de Conj_Exemplos que
                              satisfazem o teste A = v_i
                        SE Conj_Exemplos_vi = ϕ
                            ENTÃO
                                 Criar uma ramificação subordinada ao novo nó com o valor de Atributo_Meta
                                     mais freqüente
                            CASOCONTRARIO
                                 INICIO1(Conj_Exemplos_vi, Atributo_Meta, Lista_Atributos – {A})
                        FIMSE
                     FIMPARA
                  FIMINICIO2
          FIMSE
   FIMSE
   Retornar raiz
FIMINICIO1
```

FIGURA 2.3 PROCEDIMENTOS PARA A CONSTRUÇÃO DA ÁRVORE DE DECISÃO [MIT97]

O passo principal de um algoritmo que constrói uma árvore de decisão é a escolha de um atributo para rotular o nó atual da árvore. Deve-se escolher o atributo que tenha o maior poder de discriminação entre as classes para os exemplos no nó atual. Para isso, deve-se utilizar uma medida de poder de discriminação de classes. A seguir são discutidas medidas baseadas na Teoria da Informação [QUI93], [COV91], as quais são usadas pelo algoritmo C4.5.

Ganho de Informação

O ganho de informação mede a redução da entropia causada pela partição dos exemplos de acordo com os valores do atributo. Ou seja, o ganho de informação representa a diferença entre a quantidade de informação necessária para uma predição correta e as

correspondentes quantidades acumuladas dos segmentos resultantes após a introdução de um novo teste para o valor de determinado atributo [EIJ99]. Para a avaliação do quanto é oportuno a introdução de um novo teste, são considerados dois momentos: primeiro, antes da inserção deste novo teste, que constitui uma nova ramificação (partições dos dados com base nos valores dos atributos [KUB98]) e, o outro, depois da sua inserção. Se a quantidade de informação requerida é menor depois que a ramificação é introduzida, isso indica que a inclusão deste teste reduz a desordem (entropia) do segmento original.

A entropia é uma medida bem-definida da desordem ou da informação encontrada nos dados. A construção de uma árvore de decisão é guiada pelo objetivo de diminuir a entropia. A introdução da entropia no processo de construção de árvores de decisão visa a criação de árvores menores e mais eficazes na classificação [WU95].

A forma de obtenção da entropia é dada por [WU95]:

- $T = PE \cup NE$ onde PE é o conjunto de exemplos positivos e NE é o conjunto de exemplos negativos;
- $p = |PE|$ e $n = |NE|$ onde $|PE|$ e $|NE|$ representam a cardinalidade de PE e NE respectivamente;
- para cada nó da árvore serão determinadas as probabilidades de um exemplo pertencente àquele nó ser um exemplo positivo ou negativo, calculadas como $p/(p+n)$ e $n/(p+n)$, respectivamente.

Assim, a entropia é definida pela quantidade de informação necessária para decidir se um exemplo pertence a PE ou a NE, segundo a expressão 2.1 [WU95]:

$$entropia\ (p,n) = -\ p\ /\ (p+n)\ \log_2 (p\ /\ (p+n)) - n\ /\ (p+n)\ \log_2 (n/(p+n)) \qquad \text{para } p \neq 0 \text{ e } n \neq 0$$

$$entropia\ (p,n) = 0 \qquad \text{caso contrário} \qquad (2.1)$$

Note-se que entropia(p,n) depende apenas de p e de n. (A expressão 2.1 assume que há apenas duas classes, mas ela pode ser facilmente generalizada para o caso de K classes, com $K > 2$.)

A entropia dos segmentos descendentes de um nó pai da árvore é acumulada de acordo com o peso de suas contribuições na entropia total da ramificação, ou seja, de acordo com o número de exemplos cobertos pela ramificação.

A métrica que é usada para escolher o melhor teste deve avaliar o "quanto de desordem será reduzido com o novo segmento e como será a ponderação da desordem em cada segmento" [BER97a].

Para avaliar o quanto de desordem é reduzido através de um novo teste, basta calcular a entropia em cada novo segmento (nó filho) criado por cada ramo, onde cada ramo é associado com um valor do atributo sendo testado.

Se o atributo X com um domínio $\{v_1, \ldots, v_N\}$ é usado como raiz da árvore de decisão, a árvore terá então N partições de T, $\{T_1, \ldots, T_N\}$, onde T_i conterá aqueles exemplos em T que possuam o valor v_i de X. Dado que T_i contém p_i exemplos de PE (positivos) e n_i exemplos de NE (negativos), a expectativa de informação requerida para a subárvore T_i é dada pela entropia(p_i,n_i) [WU95].

A medida de ganho de informação, ganho(X), obtida pela partição associada com o atributo em X, é dada pela expressão 2.2:

$$\text{ganho}(X) = \text{entropia}(p,n) - \text{entropia\_ ponderada } (X) \tag{2.2}$$

onde a entropia ponderada de X é dada pela expressão 2.3:

$$\text{entropia\_ponderada}(X) = \sum_{i=1}^{N} ((p_i+n_i) / (p+n))\text{entropia}(p_i,n_i) \tag{2.3}$$

onde N é o número de partições (segmentos) criadas pelo teste.

Taxa de Ganho

Outra medida do poder de discriminação (entre classes) de um atributo é a taxa de ganho de informação. Esta medida é definida pela expressão 2.4 [QUI93]:

$$\text{taxa\_ganho}(X) = \text{ganho}(X) / \text{informação\_corte}(X) \tag{2.4}$$

onde ganho(X) é definido pela expressão 2.2 e

$$\text{informação-corte}(X) = - \sum_{i=1}^{N} (|T_i| / |T|) * \log_2 (|T_i| / |T|) \tag{2.5}$$

informação_corte (X) é a quantidade de informação em potencial associada com o fato de um teste do atributo X particionar T em N subconjuntos.

Observações sobre ganho de informação e taxa de ganho

Segundo Quinlan [QUI93] o critério ganho de informação, apesar de apresentar bons resultados, tem um *bias* que beneficia os testes com muitas saídas (isto é, atributos com muitos valores). Este problema pode ser corrigido através da normalização deste ganho aparente atribuído ao teste com várias saídas.

O critério taxa de ganho expressa a proporção de informação gerada pela ramificação que parece ser útil para o processo de classificação.

Se a ramificação for trivial (no sentido que cada ramo é associado com apenas um exemplo), o valor informação-corte será muito pequeno e a taxa de ganho será instável. Para evitar esta situação, o critério taxa de ganho seleciona um teste que maximize o seu próprio valor, sujeito à restrição que o teste escolhido tenha um ganho de informação pelo menos maior que a média de ganho de informação sobre todos os testes avaliados [QUI93].

O C4.5 examina todos os atributos previsores candidatos, escolhe o atributo X que maximize a taxa de ganho(X) para rotular o nó atual da árvore, e repete o processo de forma recursiva para dar continuação à construção da árvore de decisão nos subconjuntos residuais $T_1, \ldots, T_N$.

Poda em Árvores de Decisão

Conforme mencionado anteriormente, geralmente uma árvore construída pelo algoritmo C4.5 deve ser podada, a fim de reduzir o excesso de ajuste (*overfitting*) aos dados de treinamento.

Existem duas possibilidades de realização da poda em árvores de decisão: parar o crescimento da arvore mais cedo (pré-poda) ou crescer uma árvore completa e podar a arvore (pós-poda). Segundo Quinlan [QUI87], "a pós-poda é mais lenta, porém mais confiável que a pré-poda".

No C4.5 foram desenvolvidos mecanismos de poda sofisticados para tratar desta questão. Um dos mecanismos de poda em árvores de decisão adotado pelo C4.5 é baseado na

comparação das taxas de estimativa de erro[2] de cada subárvore e do nó folha. São processados sucessivos testes a partir do nó raiz da árvore, de forma que, se a estimativa de erro indicar que a árvore será mais precisa se os nós descendentes (filhos) de um determinado nó *n* forem eliminados, então estes nós descendentes serão eliminados e o nó *n* passará a ser o novo nó folha.

Regras de Produção

Apesar de serem, a princípio, uma forma de representação de conhecimento intuitiva pelo usuário, em alguns casos as árvores de decisão crescem muito, o que aumenta a dificuldade de sua interpretação [QUI93]. Para combater esse problema, alguns algoritmos transformam as árvores de decisão em outras formas de representação, tais como as regras de produção.

Essa transformação é simples. Basicamente, cada percurso da árvore de decisão, indo desde o nó raiz até um nó folha, é convertido em uma regra, onde a classe do nó folha corresponde à classe prevista pelo conseqüente (parte "então" da regra) e as condições ao longo do caminho correspondem às condições do antecedente (parte "se" da regra).

As regras de produção que resultam da transformação de árvores de decisão podem ter as seguintes vantagens:

- são uma forma de representação do conhecimento amplamente utilizada em sistemas especialistas;
- em geral são de fácil interpretação pelo ser humano;
- em geral melhoram a precisão preditiva pela eliminação das ramificações que expressam peculiaridades do conjunto de treinamento que são pouco generalizáveis para dados de teste.

Na classificação, as regras identificam as definições ou as descrições dos conceitos de cada classe [HOL94].

---

[2] Pode-se definir a taxa de estimativa de erro da seguinte forma: se *N* exemplos são cobertos por determinado nó folha e *E* dentre estes *N* são classificados de forma incorreta, então a taxa de estimativa de erro desta folha é *E / N* [BER97b].

O conjunto de regras pode ser usado para descrever a relação entre os conceitos (ou classes) e as propriedades (ou atributos previsores). Um conjunto de regras consiste de uma coleção de declarações do tipo se... então ..., que são chamadas de regras de produção ou simplesmente regras. O antecedente da regra corresponde a uma descrição de conceito, e o conseqüente da regra especifica a classe prevista pela regra para os exemplos que satisfazem a respectiva descrição de conceitos.

É importante que as regras sejam acompanhadas de medidas relativas à sua precisão (ou confiança) e a sua cobertura.

A precisão informa o quanto a regra é correta: ou seja, qual a porcentagem de casos que, se o antecedente é verdadeiro, então o conseqüente é verdadeiro.

Uma alta precisão indica uma regra com uma forte dependência entre o antecedente e o conseqüente da regra.

2.3 APRENDIZADO BASEADO EM INSTÂNCIAS

Em essência, aprendizado baseado em instâncias (*Instance-Based Learning* – IBL) é um paradigma que utiliza dados armazenados ao invés de um conjunto de regras induzidas para classificar novos exemplos. A classificação de um novo exemplo é baseada na classe do(s) exemplo(s) mais similar(es), de acordo com uma determinada métrica de distância [AHA91], [FRE98].

A idéia básica do paradigma IBL é mostrada na Figura 2.4. Como indicado na figura, um novo exemplo a ser classificado é comparado com todos os exemplos armazenados (conjunto de exemplos que estão sendo minerados), e uma medida de distância (o inverso da similaridade) é computada entre o novo exemplo e os exemplos armazenados. Como resultado do cálculo da distância são selecionados, dentre os exemplos armazenados, aqueles com a distância mínima (operador MIN). O(s) exemplo(s) selecionado(s) constitui(em) a saída do algoritmo. Na tarefa de classificação, a classe do(s) exemplo(s) selecionado(s) é(são) usada(s) para predizer a classe do novo exemplo. Tipicamente, atribui-se ao novo exemplo a classe da maioria entre os exemplos selecionados.

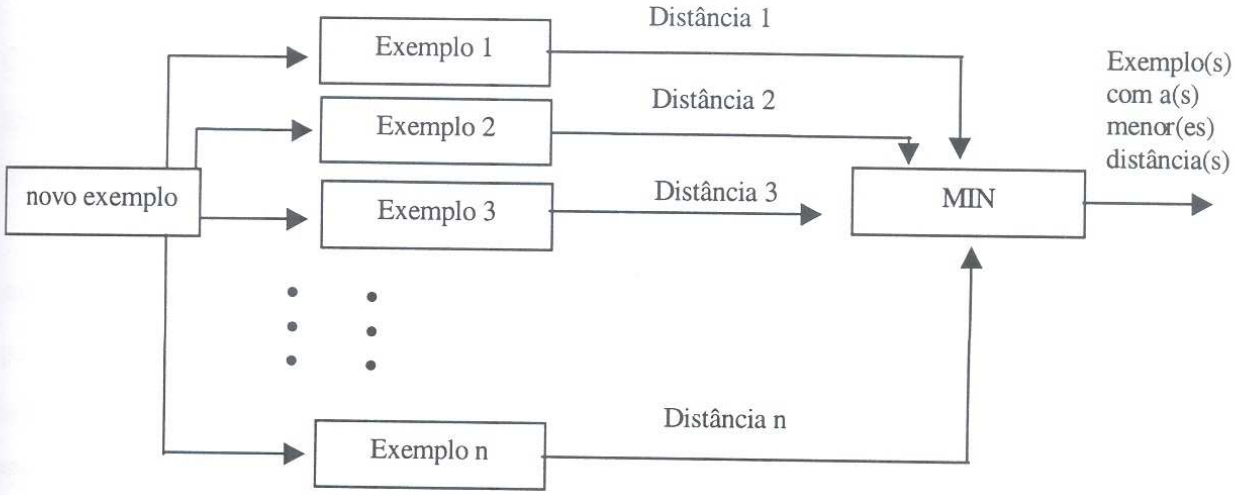


FIGURA 2.4 A IDÉIA BÁSICA DO PARADIGMA IBL [FRE98].

Quando a métrica de distância é computada, pesos de atributos podem ser usados para indicar a relevância de cada atributo na predição da classe do novo exemplo [WET95], [AHA98]. Este procedimento torna o algoritmo IBL menos sensível a atributos irrelevantes, ao adotar um menor peso para atributos irrelevantes e um maior peso para atributos relevantes. Além disso, algumas vezes a cada exemplo armazenado é atribuído um peso, o qual indica sua relevância para classificação [ATK97].

A simplicidade do paradigma IBL é uma de sua vantagens. Aliada à sua natureza não linear, os algoritmos IBL se adaptam com maior facilidade a algumas dificuldades da tarefa de predição, onde a classe do exemplo depende de uma interação entre o número de atributos previsores.

Por outro lado, algoritmos IBL têm a desvantagem de não descobrir conhecimento compreensível expresso em um alto nível de abstração, já que algoritmos IBL normalmente não descobrem regras generalizando os dados.

2.4 ALGORITMOS GENÉTICOS

O Algoritmo Genético foi concebido por John Holland em 1960 e aperfeiçoado por Holland, seus estudantes e colegas da Universidade de Michigan nas décadas de 1960 e 1970. Ao contrário da Programação Evolutiva e Estratégias de Evolução, o objetivo original de Holland não foi projetar algoritmos para problemas específicos, mas estudar como o fenômeno da adaptação ocorre na natureza e como este mecanismo poderia ser introduzido nos sistemas computacionais.

No mundo real, o processo de seleção natural controla a evolução de seres vivos. Organismos mais adaptados ao seu ambiente tendem a viver tempo suficiente para se reproduzirem, enquanto organismos menos adaptados, em geral, morrem antes de se reproduzirem.

Algoritmos Genéticos (AG) são algoritmos de busca baseados no mecanismo da seleção natural e na genética. Eles se baseiam na sobrevivência da melhor solução candidata para um determinado problema. As soluções candidatas são normalmente representadas por indivíduos artificiais. Neste trabalho a representação do indivíduo será o antecedente de uma regra de classificação do tipo se ... então, conforme será mostrado posteriormente. A cada nova geração um novo conjunto de indivíduos artificiais é criado usando segmentos (partes) dos melhores indivíduos da geração anterior, conforme avaliado por uma função de *fitness* (função de avaliação) [FAL98].

Um AG difere dos algoritmos de busca tradicionais principalmente nos seguintes aspectos [GOL89]:

- AGs realizam uma busca usando uma população de pontos (soluções candidatas), e não um único ponto.
- AGs usam operadores probabilísticos e não operadores determinísticos.
- AGs usam diretamente a informação de uma função objetivo (cujo valor ótimo deseja-se encontrar), e não derivadas ou conhecimento auxiliar sobre o problema.

A primeira e a segunda característica citadas anteriormente contribuem para a robustez de AGs, ajudando-os a escaparem de pontos de extremos locais, a fim de alcançar o ponto de extremo global. A terceira característica contribui para a generalidade de AGs que podem ser aplicados em muitos tipos de problemas.

Os principais mecanismos de um algoritmo genético simples são fáceis de entender, e não envolvem nada mais complexo do que cópias, trocas parciais entre indivíduos e pequenas alterações de elementos dos indivíduos.

A seguir são listados alguns procedimentos que geralmente são comuns em sistemas que utilizam algoritmos genéticos [BER97a]:

1. Definir o problema a ser resolvido – codificar uma solução candidata em um indivíduo artificial, e especificar uma função de avaliação.
2. Inicializar a população – tipicamente atribuindo valores aleatórios aos indivíduos da população inicial.

3. Permitir a seleção dos mais aptos e extinção dos menos aptos.
4. Permitir que uma nova geração seja formada aplicando-se processos de seleção, cruzamento e mutação aos indivíduos da geração anterior.
5. Parar quando algum(ns) do(s) seguintes critérios for(em) atingido(s):
   5.1 A solução é boa o suficiente.
   5.2 O sistema atingiu o número determinado de gerações.
   5.3 O sistema não consegue mais evoluir.

   Senão retornar ao passo 3.

2.4.1 Codificação do indivíduo e função de *fitness*

Um algoritmo genético trabalha com a representação de soluções candidatas, isto é, cada indivíduo é uma representação de um ponto do espaço de busca, dentre todas as soluções possíveis de um problema. A codificação do indivíduo é o mapeamento de um ponto do espaço de busca para um conjunto de genes formando um genoma.

A maneira como as soluções candidatas são representadas é de fundamental importância no sucesso dos métodos de busca. Muitos algoritmos genéticos utilizam representação binária, com cadeias de tamanho fixo, devido a diversas razões. Uma dessas razões é histórica: Holland e seus colegas concentraram-se nesta forma de representação e produziram a maioria da teoria existente sobre algoritmo genético considerando cadeias binárias de tamanho fixo. Mas a codificação binária não é uma representação natural para a maioria dos problemas, e novas formas foram propostas, sendo que, em diversos casos a representação usando múltiplos caracteres ou valores reais apresentaram melhor desempenho [MIC96].

No contexto de *Data Mining* a codificação de um indivíduo é, em geral, uma seqüência linear de condições de regras, sendo usualmente cada condição um par atributo-valor.

A codificação do indivíduo e a função de *fitness* influenciam diretamente nos resultados obtidos pelo algoritmo genético. A função de *fitness* é responsável pela avaliação do quão bom é o indivíduo. Todo o processo de seleção é diretamente influenciado pela função de *fitness*.

2.4.2 Métodos de seleção

A seleção direciona o processo de busca para regiões mais promissoras do espaço de busca. Um método de seleção é um processo no qual escolhe-se quais indivíduos serão submetidos aos operadores genéticos para gerar a próxima geração, o que ocorre de forma probabilisticamente proporcional aos valores da função de *fitness f*. Intuitivamente, pode-se pensar em *f* como uma medida de qualidade de solução que se quer maximizar. Os indivíduos que tiverem maior valor de *fitness* terão maior probabilidade de contribuir com um ou mais descendentes para a próxima geração[DEB00].

O processo de seleção pode ser implementado de diversas maneiras, tais como [BAC91], [SAR00], [FOG00]:

roleta – neste método a probabilidade de um indivíduo ser escolhido para reprodução é diretamente proporcional ao seu valor de *fitness*, em relação à soma dos valores de *fitness* de todos os indivíduos da população. Trata-se de um método que além de trabalhar só com valores positivos de *fitness*, é fortemente dependente da distribuição da *fitness* entre os indivíduos [BLI95].

torneio - consiste em obter aleatoriamente *K* indivíduos da população, e aquele que apresentar o maior valor de *fitness* é selecionado para a reprodução. *K* é o tamanho do torneio, um parâmetro definido pelo usuário. Em geral, quanto maior o valor de K, maior é a pressão seletiva, ou seja, mais rapidamente um indivíduo forte dominará a população e indivíduos fracos serão extintos.

2.4.3 Estratégia Elitista

O ciclo de criação e extinção dos indivíduos está diretamente relacionado à forma de gerenciamento da população. O tempo de vida de um indivíduo é tipicamente determinado por uma geração. Mas em algumas implementações de sistemas evolucionários este tempo pode ser maior. A estratégia elitista relaciona o tempo de vida de um indivíduo à sua respectiva *fitness*, ou seja, ela mantém boas soluções na população por mais de uma geração [SAR00]. Essa estratégia pode ser usada em combinação com outro método de seleção. Nesta estratégia são selecionados os *N* melhores indivíduos da geração atual para a próxima geração, onde *N*, o fator de elitismo, é um número (geralmente pequeno) definido pelo usuário.

2.4.4 Operadores genéticos

Um algoritmo genético simples é constituído de dois operadores genéticos básicos: cruzamento e mutação [GOL89]. A seguir são descritas brevemente algumas versões desses operadores.

Cruzamento (*crossover*)

O cruzamento permite a troca de material genético já disponível na população [BAC94]. Um cruzamento (*crossover*) simples pode ser feito em dois passos. Após serem escolhidos dois indivíduos, chamados pais, a partir do método de seleção, uma posição p é selecionada como um número aleatório entre 1 e n-1, onde n é o número de genes (ou características) que compõem um indivíduo. (Nos operadores de cruzamento descritos nesta seção, assume-se que ambos os indivíduos têm o mesmo número de genes.) Segundo, os genes entre a posição p+1 e n, inclusive, são trocados entre os dois pais para produzir dois novos indivíduos, chamados filhos [BOO00], [PAW95]. Este método é chamado de cruzamento em ponto simples ou cruzamento em um ponto (Figura 2.5, parte superior).

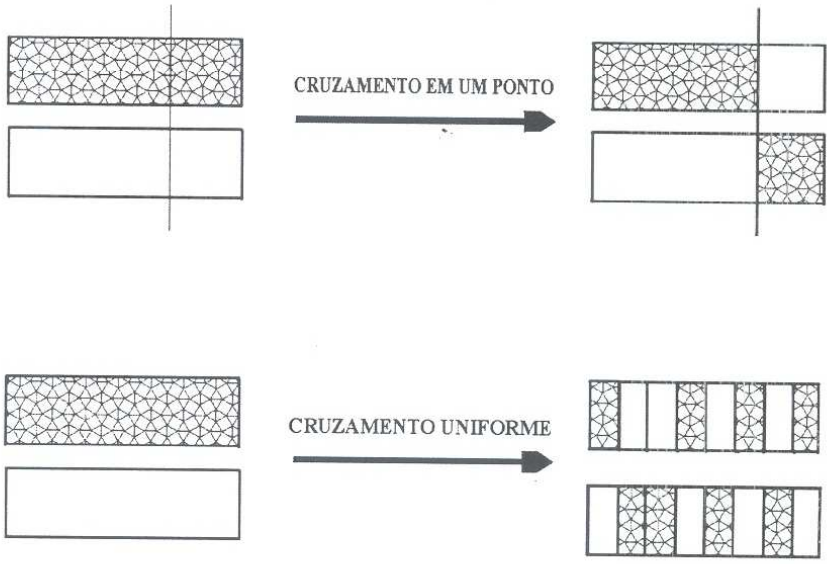


FIGURA 2.5 TIPOS DE CRUZAMENTO

Outro tipo de cruzamento possível é o cruzamento uniforme [SYS89], onde todos os genes têm a mesma probabilidade de serem trocados, independentemente de suas posições no genoma (Figura 2.5, parte inferior).

Em geral existe uma taxa de cruzamento que controla a freqüência com que os cruzamentos ocorrerão.

Mutação

O operador genético de mutação introduz inovações no material genético da população, e isto ocorre de forma aleatória [BAC94]. Em algoritmos genéticos existem vários modos de implementar o operador mutação. Por exemplo, se o indivíduo consiste de genes binários, a mutação consiste em inverter o valor de um bit; se o indivíduo consiste de genes que representam condições de regras (atributo, operador e valor), a mutação pode alterar o operador, bem como o valor, etc.

Note que a mutação pode introduzir material genético que não está presente em nenhum indivíduo da população; ao contrário do cruzamento, que apenas troca material existente entre dois indivíduos. Assim, a mutação contribui para aumentar a diversidade genética da população.

Em geral, existe uma taxa de mutação que controla a freqüência com que as mutações ocorrerão. Normalmente, a taxa de mutação é bem menor do que a taxa de cruzamento.

O compromisso entre sobrevivência do mais apto e diversidade genética

Um aspecto importante que deve ser levado em consideração quando se está modelando um problema para ser resolvido com algoritmos genéticos é a relação entre o quanto do espaço de busca se deseja explorar e quanto de esforço deve ser gasto no aproveitamento de cada local explorado [RYA95].

Em um extremo tem-se a sobrevivência apenas do mais forte, onde apenas o mais apto será reproduzido na próxima geração. No outro extremo está a seleção aleatória dos indivíduos que poderão reproduzir. No primeiro caso tem-se uma valorização extrema de uma região local, às custas de uma rápida perda de diversidade genética da população, causando uma convergência prematura para uma solução subótima. No segundo caso tem-se uma valorização extrema da exploração do espaço de busca, às custas de se perder a oportunidade de reproduzir bons indivíduos, levando à falta de convergência para uma boa solução.

Epistasia

Um sistema onde um gene afeta o significado de outro é chamado de epistático. Quando a epistasia está presente, não é possível fixar um gene e então variar o outro. Os genes, que constituem um indivíduo devem ser avaliados como um todo. Existem vários problemas em que a avaliação individual dos genes leva a resultados complemente opostos aos desejados [BER97a]. No contexto de *Data Mining*, assumindo que genes correspondem a condições de regras de classificação, o problema de epistasia corresponde ao problema de interações entre atributos, o qual é discutido por exemplo em [FRE01].

2.4.5 Abordagens de Michigan e de Pittsburgh

De modo geral, existem dois métodos de AGs para descoberta de regras [SMI00]. Em um deles cada membro da população do AG representa um conjunto completo de regras para o problema em questão. Este tipo de abordagem é chamada de abordagem de Pittsburgh.

Na outra abordagem cada membro da população representa uma única regra. Este tipo de abordagem é identificada como abordagem de Michigan. O método proposto neste trabalho adota a abordagem de Michigan.

2.4.6 Nichos

Conforme mencionado anteriormente, os Algoritmos Genéticos são baseados no princípio da seleção natural e na genética, operando de forma análoga à evolução natural. Entretanto, enquanto o processo evolucionário natural mantém uma variedade de espécies, cada uma ocupando um nicho ecologicamente separado, a população de um Algoritmo Genético tradicional rapidamente converge para uma população uniforme, ou próxima de estar uniforme, contendo soluções similares a uma única solução.

Uma forma de tratar esta questão é modificar o AG para considerar a competição por recursos finitos e limitados (nichos), resultando na criação das espécies em cada nicho [MIC96], [HOR97]. Assim, um nicho pode ser visto como um recurso do ambiente, e uma espécie é uma subpopulação que explora um nicho.

Os métodos de *niching* permitem que os AGs mantenham uma população de indivíduos diversa. AGs que incorporam métodos de *niching* são capazes de localizar múltiplas soluções de boa qualidade e mantê-las em uma mesma população. As técnicas de

niching são importantes para o sucesso dos AGs em classificação, aprendizado de máquina, otimização de funções multimodais, otimização de funções multiobjetivos e simulação de sistemas adaptativos e complexos [MAH95].

Inúmeros métodos têm sido implementados para induzir nichos e espécies em AGs. Entretanto existem pelo menos dois problemas identificados quando a questão é *niching*, que são: um modelo de abstração simples do nicho e a formação/ manutenção das espécies.

Na natureza, quando um ambiente torna-se "saturado" por um determinado tipo de organismo, indivíduos são forçados a compartilhar os recursos disponíveis. Sendo assim, pode-se concluir que a necessidade de compartilhamento é uma conseqüência natural de ambientes contendo superpopulação e conflitos.

A seguir é apresentada uma visão geral de 4 métodos de niching em algoritmos genéticos, incluindo aplicações, para descoberta de regras, no contexto de *Data Mining* e aprendizado de máquina.

Cabe ressaltar que em alguns destes métodos de *niching* o processo de compartilhamento de recursos ocorre indiretamente e em outros de forma explícita.

Fitness Sharing

Este método foi desenvolvido por Goldberg e Richardson [GOL87].

Segundo Mahfoud [MAH95] trata-se de um método de *niching* que tem sido aplicado para classificação por vários autores. *Fitness Sharing* trabalha através da redução do valor de *fitness* de elementos similares na população.

Fitness sharing embute uma pressão para a diversidade da população através da função de *fitness*. No cálculo da função de *fitness* os indivíduos não podem usar o seu valor de *fitness* sozinho, eles devem compartilhá-lo com os demais indivíduos que estejam próximos no espaço de busca. Especificamente, a *fitness* recalculada para determinado indivíduo é igual a sua *fitness* original dividida por sua contagem de nicho (*niche count*). A contagem de nicho de um indivíduo é a soma dos valores da função de *sharing* entre ele e cada indivíduo na população (incluindo ele mesmo). A função de *sharing* é uma função entre dois elementos da população, a qual retorna 1 se os elementos são idênticos, 0 se eles ultrapassam algum *threshold* de dissimilaridade; e um valor intermediário para níveis intermediários de similaridade. O *threshold* de dissimilaridade é especificado pela constante σ_{share}, a qual indica a distância máxima permitida para um *sharing* distinto de zero ocorrer [MAH93].

A analogia com um problema de otimização (maximização) é que a localização de cada ponto de máximo representa um nicho, e a capacidade de compartilhar a *fitness* associada a cada nicho pode encorajar a formação de subpopulações estáveis em cada máximo. *Fitness sharing* é baseado no princípio que o *payout* (valor) da *fitness* dentro do nicho é finito e deve ser compartilhado entre todos os indivíduos ocupando aquele nicho. Consequentemente, a *fitness* atribuída a um indivíduo será inversamente proporcional ao número de outros indivíduos no mesmo nicho.

Note que *fitness sharing* é um método computacionalmente caro, que requer a computação de N^2 medidas de distância entre pares de indivíduos, onde N é o número de indivíduos da população.

A complexidade do algoritmo pode ser reduzida para $O(Np)$ - onde N é o número de indivíduos e p é a proporção de picos de interesse - se for adotada uma amostra da população, ao invés de calcular a distância entre cada par de indivíduos. Essa sugestão, que também já havia sido feita por Goldberg e Richardson [GOL87], foi implementada em Goldberg, Horn e Deb [GOL92], e tem sido mencionada como importante por outros autores [BEA93].

Desvantagens: Segundo Goldberg e Wang [GOL97] um dos pontos mais críticos deste método é a necessidade da especificação do valor do σ_{share}. Uma estimativa precisa de σ_{share} requer o conhecimento do número de picos no espaço. Para solucionar este problema algumas soluções alternativas podem ser adotadas, como, por exemplo, não adotar um σ_{share} fixo [BEA93]. Outra desvantagem de *fitness sharing* é que computar precisamente a *fitness* de um indivíduo envolve o cálculo de sua distância em relação a todos os outros membros da população [SMI93]. A complexidade de tempo total será dependente do tempo necessário para o AG básico, mais o tempo adicional para processar os cálculos de *fitness sharing*. Este adicional tem como complexidade $O(N^2)$ (N é o tamanho da população). Porém, essa desvantagem pode ser reduzida usando-se amostras da população para cálculo da conta de nicho de cada indivíduo, como mencionado anteriormente.

Vantagem: apesar das limitações citadas, *fitness sharing* é importante pela indução de nichos estáveis. *Fitness sharing* tende a espalhar a população sobre múltiplos picos (nichos) em proporção à altura dos picos.

COGIN

COGIN [GRE93] - *COverage-based Genetic Induction* – é um algoritmo de indução de regras baseado em competição.

Trata-se de um sistema de aprendizado que utiliza um método de *niching* implícito, ou seja, não existe nenhum parâmetro de *niching*.

A novidade apresentada por este algoritmo reside no fato dele usar o conjunto de treinamento como uma restrição explícita para a complexidade (tamanho) de um conjunto de regras, e criar uma pressão seletiva para que a diversidade no conjunto de regras seja mantida. A sobrevivência de um indivíduo está diretamente relacionada à sua habilidade de preencher um determinado nicho no espaço de dados.

Este sistema trabalha com regras (indivíduos) de tamanho fixo, mas com uma população de tamanho variável.

Novas regras candidatas são geradas por cruzamento de regras na população. COGIN usa cruzamento de ponto único. Após cruzamento, todo o conjunto de regras candidatas é ordenado pelo valor da *fitness*. O conjunto de treinamento é submetido a este conjunto ordenado de regras. Mais precisamente, o conjunto de treinamento é inicialmente submetido à primeira regra (de melhor *fitness*). Os exemplos corretamente cobertos por essa regra são removidos do conjunto de treinamento. Os exemplos restantes são submetidos à próxima regra (com a segunda melhor *fitness*), e assim por diante. Quando todos os exemplos tiverem sido tratados, o sistema não precisa de mais regras. O subconjunto de regras que restarem no conjunto de regras pode ser eliminado, e um novo modelo (população) é estabelecido.

O modelo é naturalmente hierárquico, contendo regras gerais dominantes e, além disso, contendo regras específicas preenchendo os espaços restantes.

Tratando os exemplos como nichos, o método provê um número mínimo de regras cobrindo os nichos, com o melhor indivíduo designando cada nicho.

O uso da cobertura de exemplos como uma restrição explícita no tamanho do modelo (população) representa uma mudança significativa em relação a abordagens anteriores para o problema de manutenção da diversidade na construção de classificadores. A maioria dos projetos com sistemas classificadores seguindo a abordagem de Michigan (os sistemas que mais se aproximam da estrutura COGIN) têm procedimentos baseados em um modelo de tamanho fixo, o qual deve ser grande o suficiente para cobrir todos os exemplos.

Vantagem: em contraste a um classificador típico, o tamanho da população irá variar ao longo da evolução, se adaptando dinamicamente aos dados [GRE93].

Desvantagem: é relativamente difícil a compreensão do conjunto de regras dada a sua forma de apresentação a partir de um conjunto de regras ordenadas, dado que, para entender a *n-ésima* regra, é preciso levar em consideração as *n-1* regras anteriores [CLA91], [NOC98].

Nicho Seqüencial

Este método de *niching* executa repetidas vezes o AG tradicional e, a cada execução, assegura que uma nova região do espaço de busca esteja sendo pesquisada.

Este método é atribuído a Beasley, Bull e Martin [BEA93]. Ele trabalha através da iteração de um AG simples e mantém a melhor solução de cada execução. Desta forma, o método constrói nichos seqüencialmente para resolver um único problema. Para impedir a convergência sobre uma mesma área no espaço de busca várias vezes, o algoritmo localiza uma determinada solução e reduz a *fitness* de todos os indivíduos que estão localizados dentro do raio do nicho. Esta medida de raio do nicho é similar ao σ_{share} no *fitness sharing*.

O método nicho seqüencial trabalha modificando a *fitness* de acordo com a localização das soluções encontradas em execuções anteriores (ao contrário do método *fitness sharing*, onde a paisagem de *fitness (fitness landscape)* se mantém estática durante cada execução). Isto é feito para desencorajar indivíduos a explorar novamente uma área onde soluções já tenham sido encontradas.

Uma vez localizado um ponto de máximo, este não precisa ser localizado novamente. Numa próxima execução é assumido que este nicho já está coberto e pouca atenção é dispensada a qualquer indivíduo que ainda esteja cobrindo esta área. Os indivíduos são forçados a convergir para um nicho ainda não coberto, o qual será também considerado coberto numa terceira execução do AG. Este processo pode continuar até que um critério preestabelecido determine que todos os máximos tenham sido localizados.

Tendo definida a função de *fitness* e a métrica de distância, o algoritmo de nicho seqüencial é [BEA93]:

a. Inicializa: comparar o valor da função de *fitness* modificada com o valor da *fitness* original
b. Executa o AG (ou outra técnica de busca), usando a *fitness* modificada, identificando o melhor indivíduo gerado pelo AG.

c. Atualiza a *fitness* modificada de forma a reduzir o valor de *fitness* na região próxima ao melhor indivíduo, produzindo uma nova *fitness* modificada.

d. Se a *fitness* original do melhor indivíduo (identificado no passo b) exceder um patamar de solução (*solution threshold*), mostrar esta como uma solução.

e. Se nem todas as soluções tiverem sido encontradas, de acordo com o critério de parada estabelecido, retornar ao passo b.

Uma única aplicação completa deste algoritmo é referenciada como uma seqüência, uma vez que ele consiste de uma seqüência de várias execuções do AG. O conhecimento da localização de um nicho (um ponto de máximo) é propagado para execuções subseqüentes na mesma seqüência.

Vantagens: Baesley [BEA93] menciona três vantagens de seu método:

• simplicidade – Trata-se de um método conceitualmente simples, se comparado aos métodos já existentes.

• habilidade para trabalhar com populações pequenas, uma vez que o objetivo durante cada execução do AG é localizar apenas um pico.

• rapidez, como uma conseqüência da segunda vantagem. Porém, cabe ressaltar que essa vantagem é parcialmente anulada (e talvez eliminada) pelo fato de que o AG deve ser executado múltiplas vezes. Segundo Mahfoud [MAH95], métodos de *niching* paralelos são mais rápidos que o seqüencial e obtêm melhores resultados, mesmo quando executados em um único processador.

Desvantagens: Segundo Mahfoud [MAH95] pode ocorrer perda da propriedade de cooperação da população.

REGAL

REGAL (*RElational Genetic Algorithm Learner*) é um sistema baseado em AG distribuído, projetado para aprendizado de conceitos multimodais representados em lógica de primeira ordem a partir de exemplos [GIO95].

A formação de espécies é obtida através do mecanismo de seleção, da mesma forma que o método que usa o conceito de *sharing*. No caso do *sharing* o mecanismo de seleção corresponde ao mecanismo básico de seleção de um AG simples, e a formação das espécies é obtida através da definição de uma função de *fitness* mais elaborada (*fitness* reduzida). A abordagem adotada pelo método REGAL é oposta. Usa uma função de *fitness*

simples, sendo o mecanismo de seleção modificado, conforme a tarefa e o domínio da aplicação. Isto significa dizer que o método REGAL é dependente da tarefa sendo resolvida. Em particular, REGAL foi desenvolvido para a tarefa de classificação.

O mecanismo de seleção é obtido através da introdução de um operador chamado de sufrágio universal. Este operador não usa nenhuma medida de distância e não requer nenhum parâmetro adicional.

Trata-se de uma abordagem bastante similar ao compartilhamento (*sharing*) de recursos no espaço fenotípico[3]. Funciona modificando o método de seleção para promover a criação de nichos e a competição entre as regras. O método de seleção do REGAL trata as regras como candidatos que concorrem pelos votos dos eleitores, representados pelos exemplos de treinamento. Cada exemplo vota em apenas uma das regras que o cobre, sendo que a escolha é realizada probabilisticamente por um processo semelhante à seleção por roleta, onde cada regra tem uma fatia da roleta proporcional à *fitness* daquela regra. Ou seja, a probabilidade de um exemplo votar em uma regra é proporcional à qualidade daquela regra. As regras com maior número de votos são selecionadas para cruzamento [HAS00a].

Vantagem: não necessita definir uma medida de distância.

Desvantagem: a escolha da regra na qual um exemplo vota é realizada de forma probabilística usando a regra da roleta, o que pode introduzir alguns problemas, conforme apresentado na seção 2.4.2.

Tabela Comparativa

A Tabela 2.1 apresenta uma síntese das principais características dos 4 métodos de *niching* discutidos no item anterior. Nessa tabela a primeira coluna identifica o método de *niching*. A segunda coluna indica se o método necessita da definição de parâmetros e quais são estes parâmetros. A terceira coluna (implícito/explícito) caracteriza se o método utiliza *niching* implícito ou explícito. O que determina que um método de *niching* seja implícito ou explícito é o fato do método ter ou não algum parâmetro específico de *niching*. Para o caso afirmativo, o método foi considerado explícito, caso contrário foi considerado implícito. A

[3] O compartilhamento de recursos no espaço fenotípico permite uma comparação mais adequada de similaridade entre 2 indivíduos, em comparação com o compartilhamento genotípico. Isto ocorre porque a similaridade é medida pelo número de exemplos de treinamento que são cobertos simultaneamente pelas regras. Quanto mais exemplos duas regras cobrem, mais os seus valores de *fitness* serão reduzidos.

quarta coluna indica como se dá o processo de *niching* (determinístico / probabilístico). A quinta coluna indica se o método foi ou não desenvolvido para o contexto da tarefa de classificação. Finalmente, a última coluna descreve a forma de obtenção da função de *fitness*.

TABELA 2.1 COMPARATIVO DOS MÉTODOS DE *NICHING*

Método	Parâmetros	Implícito/ Explícito	*Niching*	Tarefa classif.	Característica da função de *fitness*
Fitness sharing	σ_{share}	Explícito	Determ.	Não	Deprecia *fitness*
COGIN	-	Implícito	Determ.	Sim	*Fitness* = soma recursos compart.
Nicho Seqüencial	Métrica de distância	Explícito	Determ.	Não	Deprecia *fitness*
REGAL	-	Implícito	Probab.	Sim	Função *fitness* simples – sofistica o método seleção

2.4.7 Algoritmos Genéticos para Descoberta de Regras

A grande maioria dos trabalhos existentes na literatura referentes à aplicação de AGs em *Data Mining* é relacionada à tarefa de classificação e, muitos desses trabalhos propõem AGs para descoberta de regras na forma se .. então. A seguir são apresentados alguns dos trabalhos mais representativos nesta área.

Janikow [JAN91] apresenta um algoritmo genético, denominado GIL, para a tarefa de classificação. Neste algoritmo cada indivíduo representa um conjunto de regras (abordagem de Pittsburgh), onde cada regra é uma conjunção de condições e cada condição especifica um ou mais valores para um determinado atributo (disjunção interna). Cada indivíduo é um conjunto de regras de tamanho fixo.

Em GIL os operadores genéticos podem atuar em três níveis: ao nível do conjunto de regras, ao nível das regras e ao nível das condições. O primeiro nível possui os seguintes operadores: operador que realiza trocas entre dois conjuntos de regras; operador que copia aleatoriamente uma regra de um conjunto para um outro; operador que adiciona uma regra que cubra um exemplo positivo não coberto por um conjunto de regras; operador que seleciona duas regras e as troca por uma regra mais especializada e operador que seleciona duas outras regras e as troca pela regra mais genérica. O segundo nível possui os seguintes operadores: operador que divide uma regra em outras regras, operador que remove condições

da regra e operador para introduzir condição. No terceiro nível: operador que remove ou adiciona um valor da condição, aumenta o domínio das condições e diminuição do domínio das condições. Logo, este é um algoritmo genético relativamente complexo, com vários operadores projetados especificamente para *Data Mining*.

De Jong [DEJ93] apresenta o GABIL, um algoritmo genético para a aquisição de regras de classificação. Seguindo a abordagem de Pittsburgh, cada indivíduo da população representa um conjunto de regras de tamanho fixo, candidatas à solução do problema, isto é, cada indivíduo é uma *string* de tamanho variável representando um conjunto de regras de tamanho fixo. Os operadores genéticos usados são a mutação aleatória de *bit* e o cruzamento em dois pontos adaptado para indivíduos de tamanho variável. Ao contrário do GIL, GABIL usa operadores genéticos simples, praticamente sem necessidade de modificar AGs convencionais.

Congdon [CON95] apresenta em sua tese de doutorado uma comparação entre AGs e outros sistemas de aprendizado de máquina no desempenho da tarefa de classificação de doenças. Nesta comparação ela contempla algoritmos genéticos, árvores de decisão, Autoclass e Cobweb. Seus resultados demonstram que AGs tiveram uma melhor *performance* em termos de habilidade descritiva, apesar das árvores de decisão também apresentarem bom desempenho.

Giordana e Neri [GIO95] apresentam o REGAL, um sistema baseado em algoritmo genético distribuído para a tarefa de classificação. São apresentadas tanto uma versão serial como paralela do sistema. Nesta seção o foco será na versão serial. Para a representação dos indivíduos foi utilizado uma abordagem híbrida entre Pittsburgh e Michigan. Cada indivíduo codifica uma solução parcial e a população como um todo é um conjunto redundante de soluções parciais. A diferença está no fato de que cada indivíduo evolui de forma independente e apenas no final uma solução completa é formada. A abordagem é híbrida pois nem cada indivíduo representa uma solução final e nem a população total também a representa. Os operadores utilizados são o cruzamento, mutação e semeadura (*seeding*). O sistema utiliza quatro tipos de cruzamento: cruzamento em dois pontos, cruzamento uniforme, cruzamento de generalização e de especialização. Os dois primeiros são iguais aos encontrados na literatura. Os dois últimos têm a finalidade de gerar regras filho que sejam generalizações ou especializações das regras pais. Logo, da mesma forma que o GIL, REGAL também usa operadores genéticos especializados para a *Data Mining*.

Outro algoritmo genético para a tarefa de classificação foi proposto por Hasse e Pozo [HAS00b]. O algoritmo segue a abordagem de Michigan, e usa *fitness sharing* para incentivar a diversidade de regras (indivíduos) na população.

O algoritmo genético GLOWER, proposto por [DHA00], procura realizar o aprendizado de regras de classificação, no caso, previsões financeiras para realização de investimentos. Os autores também levantam as vantagens do algoritmo genético em comparação com outros métodos de busca local (árvores de decisão, indução de regras). O algoritmo realiza a descoberta de regras utilizando a técnica de criação seqüencial de nichos, revista na seção 2.4.6.

Liu e Kwok [LIU00a] propõem um algoritmo genético baseado no SIA [VEN93]. O SIA é um algoritmo baseado na idéia de dividir-para-conquistar, isto é, ele procura descobrir uma regra por vez, reduzindo assim o espaço de busca. O algoritmo proposto por Liu e Kwok é uma extensão do SIA. São apresentadas melhorias no modo de inicialização da população inicial, alterações nos operadores genéticos e modo de filtragem das regras. O ESIA (Extended SIA) gera sua população inicial garantindo que cada regra cubra pelo menos um exemplo. A mutação é utilizada como meio de manter a diversidade genética, enquanto o operador de especialização escolhe, de forma aleatória, atributos a serem especializados nos indivíduos. Os novos filtros de regras introduzidos procuram eliminar regras consideradas ruído (cobrindo mais exemplos de outras classes do que a classe prevista pela regra), regras redundantes e regras altamente incompletas (cobrindo um número inferior a um número preestabelecido de exemplos).

Papagelis e Kalles [PAP01] exploram o uso de AGs diretamente na construção de árvores de decisão binárias, com o objetivo de gerar árvores mais precisas, bem como mais simples. Os resultados apontam que os AGs têm grande vantagem sobre outras "heurísticas gulosas"(*greedy*), especialmente quando o domínio apresenta atributos irrelevantes ou fortemente dependentes.

Note que, em geral, os AGs discutidos anteriormente foram projetados para a tarefa de classificação. A principal diferença entre esses AGs e método proposto neste trabalho é o fato de este último ser um híbrido árvore de decisão/AG, projetado para tratar de um problema específico, a saber o problema de pequenos disjuntos. Nenhum dos AGs discutidos nesta seção foi projetado para resolver esse problema.

3 MÉTODO PROPOSTO

Segundo [ANA98] existe uma área de pesquisa promissora em *Data Mining* para a construção de sistemas híbridos, os quais têm a vantagem de vencer as limitações que cada um de seus paradigmas apresentam quando utilizados de forma isolada.

Desta forma, este trabalho propõe um método híbrido árvore de decisão/algoritmo genético para descoberta de regras que trata do problema de pequenos disjuntos. A idéia básica é que os exemplos pertencentes aos grandes disjuntos sejam classificados pelas regras produzidas por um algoritmo de árvore de decisão, enquanto que os exemplos pertencentes aos pequenos disjuntos (cujo processo de classificação é consideravelmente mais difícil) sejam classificados pelas regras descobertas por algoritmo genético especificamente projetado para descobrir regras para pequenos disjuntos.

Esta abordagem combina características favoráveis de ambas as técnicas de descoberta de conhecimento. Os algoritmos de árvore de decisão têm um *bias* privilegiando a generalidade das regras descobertas. Esse *bias* se adequa bem aos grandes disjuntos, mas não aos pequenos disjuntos. Em particular, um dos gargalos dos algoritmos de construção de árvore de decisão é o problema de fragmentação [FRI96], onde o conjunto de exemplos pertencentes ao nó da árvore torna-se cada vez menor à medida que a profundidade da árvore é aumentada, tornando difícil a indução de regras confiáveis a partir de níveis mais profundos da árvore.

Por outro lado, os algoritmos genéticos são métodos de busca robustos e flexíveis, que tendem a tratar bem com interações entre atributos [FRE01], [DHA00], [NOD99]. Desta forma, intuitivamente eles podem ser mais facilmente adaptados para tratar com os pequenos disjuntos, que são associados com um alto grau de interação de atributos [REN90], [NAZ99].

As razões pelas quais os AGs, e os algoritmos evolucionários em geral, tendem a tratar bem a questão da interação entre atributos é devido à sua natureza de busca global [FRE01], [GOL89]. Primeiramente, os AGs trabalham com uma população de soluções candidatas (indivíduos), ao invés de contemplar uma única solução por vez (como a maioria dos algoritmos de indução de regras faz). A segunda razão é pela forma como é feita a avaliação. No AG uma solução candidata é avaliada, como um todo, através da função de *fitness*. Esta característica contrasta com a maioria dos algoritmos de indução de regras, onde o procedimento de busca avalia uma solução candidata parcial, baseado unicamente em informação local. A terceira razão é o fato dos AGs utilizarem operadores probabilísticos. Isso

tende a evitar que a busca se restrinja a um espaço em torno de um ponto de máximo local, ou seja, ajuda a busca a escapar de máximos locais, aumentando a chance de encontrar um máximo global.

O sistema proposto neste trabalho para a descoberta de regras de classificação tem duas fases de treinamento (Figura 3.1).

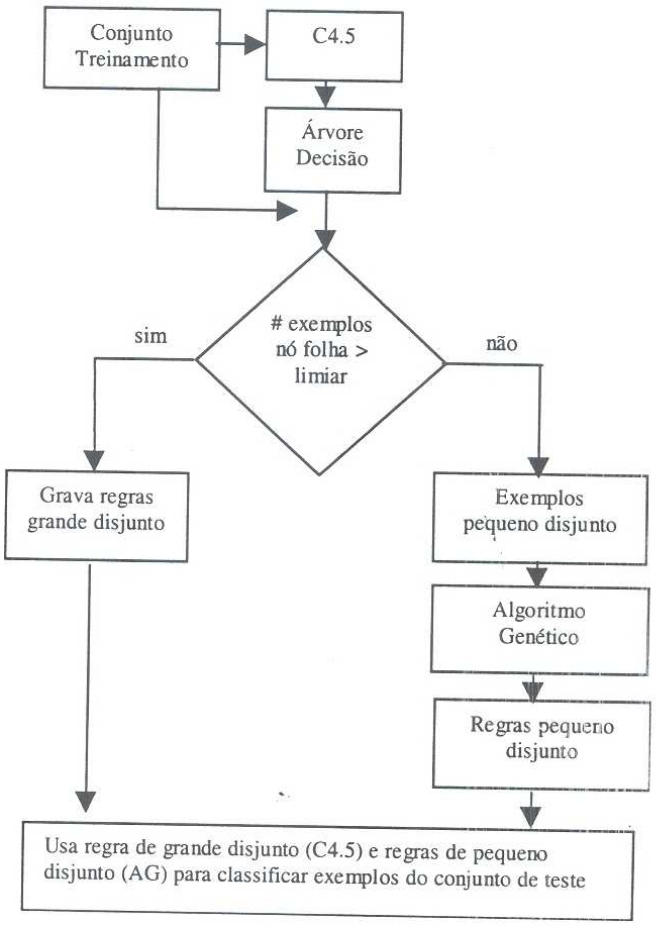


FIGURA 3.1 VISÃO GERAL DO MÉTODO HÍBRIDO ÁRVORE DE DECISÃO / AG

Na primeira fase é executado o algoritmo C4.5, um algoritmo bastante utilizado para indução de árvore de decisão [QUI93]. Cabe lembrar que esse algoritmo foi descrito na seção 2.2.

A árvore induzida pelo C4.5, na sua forma podada, é transformada em um conjunto de regras, na forma usual – a saber, cada percurso compreendido entre o nó raiz e um nó folha corresponde a uma regra que prediz a classe especificada por aquele nó folha. Assim, uma árvore de decisão com d nós folhas é transformada em um conjunto com d regras (ou disjuntos). Cada uma destas regras é considerada como um pequeno disjunto ou como um grande (não-pequeno) disjunto, dependendo do número de exemplos cobertos pela regra. Se

esse número for menor ou igual a um determinado valor predefinido (conforme discutido na seção 4.3), a regra é considerada um pequeno disjunto. Caso contrário a regra é considerada um grande disjunto.

A segunda fase consiste em usar um algoritmo genético para descobrir regras que cubram os exemplos pertencentes aos pequenos disjuntos. Este trabalho propõe dois algoritmos genéticos para cumprir esta fase, que serão descritos nas duas próximas seções.

3.1 ALGORITMO GENÉTICO PARA DESCOBERTA DE REGRAS PARA CADA PEQUENO DISJUNTO (AG-PEQUENO)

A idéia básica deste algoritmo genético (AG), identificado como AG-Pequeno, é que a cada execução do AG sejam descobertas regras que classifiquem exemplos pertencentes a um pequeno disjunto em separado, ou seja, exemplos pertencentes a um nó folha da árvore de decisão que tenha sido categorizado como pequeno disjunto. Assim, cada execução do AG usará um pequeno conjunto de treinamento, contendo exemplos de apenas um pequeno disjunto [CAR00a], [CAR00b].

O primeiro procedimento em projetar um AG que descubra regras é decidir qual é a representação de um indivíduo (solução candidata) da população. No caso deste trabalho, cada indivíduo representa uma regra de pequeno disjunto. O genoma de um indivíduo consiste das condições que compõem o antecedente (parte "se") da regra. O objetivo do AG é construir condições que maximizem a precisão preditiva da regra, a qual é avaliada pela função de *fitness* – descrita no item 3.1.2. O AG também possui um operador de poda que favorece a descoberta de regras menores, mais simples – descrito no item 3.1.4.

O conseqüente (parte "então") da regra, que especifica a classe predita, não é representado no genoma. Na verdade, ele é fixo para cada execução do AG. Desta forma, todos os indivíduos de uma mesma execução do AG representam uma regra com o mesmo conseqüente.

Cada execução do AG descobre uma única regra prevendo uma certa classe para os exemplos pertencentes a um dado pequeno disjunto. Esta única regra é a melhor regra encontrada dentre todas as regras geradas ao longo de todas as gerações. Uma vez que é necessário construir regras que prevejam todas as classes nos diversos pequenos disjuntos, se torna imperativo executar o AG várias vezes para uma mesma base de dados. Mais precisamente, são executadas $d * c$ vezes o AG, onde d é o número de pequenos disjuntos e c

é o número de classes a serem preditas. Para um dado pequeno disjunto, a *i-ésima* execução do AG, $i = 1, \ldots, c$, descobre a regra previsora da *i-ésima* classe.

3.1.1 Representação do indivíduo

Conforme mencionado anteriormente, neste trabalho cada indivíduo da população do AG representa o antecedente (parte SE) de uma regra de pequeno disjunto. Mais precisamente, cada indivíduo representa uma conjunção de condições compondo o antecedente de uma dada regra. Cada condição é um par atributo-valor, como será descrito em mais detalhes posteriormente.

O antecedente da regra contém um número variável de condições, uma vez que não se tem a *priori* a informação de quantas condições serão necessárias para compor uma regra de boa qualidade. Na prática, para a implementação, é preciso especificar o número mínimo e o máximo de condições que podem ocorrer no antecedente da regra. Para a implementação neste trabalho, foi determinado 2 (dois) como sendo o número mínimo. Esse número mínimo foi determinado considerando que um antecedente de regra contendo uma única condição provavelmente seria insuficiente para indicar a classe de um exemplo de um pequeno disjunto. (Note que, se fosse possível classificar corretamente exemplos de pequeno disjunto com uma regra contendo uma única condição, provavelmente o algoritmo de árvore de decisão já teria estendido a árvore para incluir o atributo daquela condição.)

O número máximo de condições na regra é mais difícil de ser determinado. A princípio, o número máximo de condições poderia ser m, onde m é o número de atributos previsores na base de dados. Entretanto, esta opção apresenta duas desvantagens. Primeiramente, pode conduzir o processo à descoberta de regras muito longas, característica que contraria o princípio desejável de que o conhecimento descoberto seja simples (conforme apresentado na seção 2.1). Segundo, requer uma representação de genoma longa para a caracterização dos indivíduos, o que leva a um aumento no tempo de processamento.

Para evitar estes problemas, foi utilizada uma heurística para selecionar um subconjunto de atributos que podem ser usados para compor as condições que representam o antecedente das regras.

Esta heurística é baseada no fato que diferentes pequenos disjuntos identificados pelo algoritmo de árvore de decisão podem ter várias condições ancestrais em comum na árvore. Por exemplo, suponha, que dois nós folha irmãos da árvore de decisão foram

identificados como sendo pequenos disjuntos, e que k é o número de nós ancestrais destes dois nós. Então os dois antecedentes de regra correspondentes têm k - 1 condições em comum. Desta forma, não faz muito sentido usar as condições comuns para compor as regras descobertas pelo AG, uma vez que, em geral, não haverá como discriminar os dois pequenos disjuntos correspondentes. Como conseqüência, para cada pequeno disjunto, o genoma do indivíduo contém apenas os atributos que não ocorrem em nenhum nó ancestral do nó folha que define aquele pequeno disjunto. Um outro motivo que colabora com esta argumentação é o fato de que, como o conjunto de treinamento a ser submetido ao AG para evolução terá muito poucos exemplos, o número de atributos previsores não deve ser elevado, para evitar que seja realizada uma busca em um espaço de dados grande demais para tão poucos exemplos [SCH93a], [PRO96], [PRO99].

Para representar o antecedente de tamanho variável de uma regra foi utilizado um genoma de tamanho fixo, a fim de simplificar a implementação, conforme explicado a seguir. Deve ser lembrado que cada execução do AG descobre uma regra associada a um determinado pequeno disjunto. Para uma dada execução do AG, o genoma de um indivíduo é composto de n genes, onde $n = m - k$, sendo m o número total de atributos previsores na base de dados e k o número de nós ancestrais na árvore de decisão que identifica o pequeno disjunto em questão.

Cada gene representa uma condição da regra na forma $A_i\ Op_i\ V_{ij}$, onde o subscrito i identifica a condição da regra, $i = 1,...,n$; A_i é o i-*ésimo* atributo; V_{ij} é o j-*ésimo* valor do domínio de A_i; e Op_i é um operador relacional compatível com o A_i (Figura 3.2). Este operador Op_i pode ser "=" ou "in" para o caso de atributos categóricos, ou "≤" ou ">", para atributos contínuos. A identificação de valores $\{V_{i1},...,V_{ij}\}$ é usada se A_i for categórico, ou enquanto um único valor V_{ij} é usado se A_i for contínuo. B_i representa um *bit* ativo, usado como *flag*, que assume valor 1 ou 0 para indicar se a respectiva i-*ésima* condição está presente ou não no antecedente da regra, respectivamente.

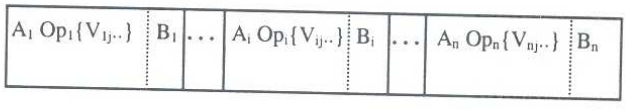


FIGURA 3.2 ESTRUTURA DO GENOMA DE UM INDIVÍDUO.

3.1.2 Função de *Fitness*

Seja classe positiva ("+") a classe predita pela regra em questão, e classe negativa ("-") qualquer outra classe distinta da classe predita pela regra.

Para avaliar a qualidade de um indivíduo (regra candidata), o AG usa a seguinte função de *fitness* [HAN97]:

$$Fitness = (VP / (VP + FN)) * (VN / (FP + VN)), \text{ onde} \quad (3.1)$$

VP (verdadeiro positivo) = número de exemplos "+" que foram corretamente classificados como exemplos "+";
FP (falso positivo) = número de exemplos "-" que foram incorretamente classificados como exemplos "+" ;
FN (falso negativo) = número de exemplos "+" que foram incorretamente classificados como exemplos "-" ;
VN (verdadeiro negativo) = número de exemplos "-" que foram corretamente classificados como exemplos "-" .

Na expressão 3.1, o termo $(VP / (VP + FN))$ é geralmente chamado de sensibilidade, enquanto o termo $(VN / (FP + VN))$ é geralmente chamado especificidade. Estes dois termos são multiplicados para forçar o AG a descobrir regras que tenham tanto alta sensibilidade quanto alta especificidade, uma vez que seria relativamente simples (mas indesejável) maximizar um dos termos pela redução do outro. É possível encontrar um maior detalhamento sobre esta medida de qualidade de regras de classificação em [HAN97], [LOP96]. Em [HAN97] essa medida é discutida independentemente de AGs, enquanto em [LOP96] ela é discutida no contexto de AGs.

Vale salientar que esta função de *fitness* não leva em consideração a simplicidade da regra. Entretanto, o AG proposto adota um operador de poda que promove a descoberta de regras curtas (compreensíveis), como apresentado em 3.1.4.

3.1.3 Especificação de Seleção, Cruzamento, Mutação e Elitismo

Foi adotado o método de torneio para a reprodução, com tamanho de torneio igual a dois [MIC96], [BLI00], conforme descrito na seção 2.4. Também foi implementado o operador de cruzamento de um ponto padrão, com probabilidade de cruzamento de 80%. A probabilidade de mutação usada foi de 1% por gene. Foi implementado o elitismo com fator um – ou seja, o melhor indivíduo de cada geração é repassado inalterado para a próxima geração.

3.1.4 Operador de poda da regra

Foi desenvolvido um operador especialmente projetado para melhorar a simplicidade das regras candidatas. A idéia básica deste operador, chamado de operador de poda, é remover várias condições da regra para torná-la menor. Em um alto nível de abstração, remoção de condições da regra é uma forma de tornar a regra mais simples, sendo um recurso bastante usado na literatura de *Data Mining* e de aprendizado de máquina (apesar dos detalhes de implementação variarem bastante entre os diversos algoritmos) [BRE97].

No método proposto o operador de poda é aplicado em todos os indivíduos da população, após cada indivíduo ter sido construído e submetido aos operadores de cruzamento e mutação.

Diferentemente dos operadores genéticos simples tradicionais, o operador de poda proposto é um procedimento mais elaborado, baseado na teoria da informação [COV91]. Este procedimento pode ser considerado como uma forma de incorporar uma heurística relacionada à classificação dentro do AG para descoberta de regra. A heurística em questão favorece a remoção de condições de regra com baixo ganho de informação, mantendo as condições com alto ganho de informação.

O operador de poda trabalha de forma iterativa. Na primeira iteração a condição com o menor ganho de informação é considerada. Esta condição é mantida na regra (ou seja, seu respectivo *bit* ativo é ajustado para 1) com probabilidade igual ao seu ganho de informação normalizado (no intervalo entre [0..1]), e é removida da regra (seu *bit* ativo é ajustado para 0) com o complemento daquela probabilidade. A seguir a condição com o segundo menor ganho de informação é considerada. Novamente, esta condição é mantida na regra com probabilidade igual ao seu ganho de informação normalizado, e é removida da regra com o complemento daquela probabilidade. Este processo iterativo é executado enquanto o número de condições ativas na regra for maior que o número especificado como mínimo de condições – neste trabalho - determinado em dois, como explicado anteriormente – e o número de iterações é menor que o número de genes (número máximo de condições na regra) n. Note que cada gene é considerado uma única vez durante o processo iterativo de poda (Figura 3.3).

Na seção 2.2.1 foi explicado como é calculado o ganho de informação de um atributo [QUI93], [COV91]. Neste trabalho usa-se uma variação daquele cálculo, onde o ganho de informação é computado ao nível de condição da regra (par atributo valor). Mais

precisamente, o ganho de informação de cada condição da regra $cond_i$, da forma $<A_i\ Op_i\ V_{ij}>$, é calculado da seguinte forma:

$$\text{ganho}(cond_i) = \text{info}(G) - \text{info}(G|cond_i), \text{ onde} \quad (3.2)$$

$$\text{info}(G) = -\sum_{j=1}^{c} (|G_j| / |T| * \log_2 (|G_j| / |T|)) \quad (3.3)$$

$$\text{info}(G|cond_i) = -[|V_i|/|T|) \sum_{j=1}^{c} ((|V_{ij}|/|V_i|) * \log_2(|V_{ij}|/|V_i|)] - [|\neg V_i|/|T|) \sum_{j=1}^{c} ((|\neg V_{ij}|/|\neg V_i|) * \log_2(|\neg V_{ij}|/|\neg V_i|)] \quad (3.4)$$

onde G é o atributo meta (atributo classe), c é o número de classes (valores de G), $|G_j|$ é o número de exemplos de treinamento tendo o *j-ésimo* valor de G, $|T|$ é o número total de exemplos de treinamento, $|V_i|$ é o número de exemplos de treinamento que satisfazem a condição $<A_i\ Op_i\ V_{ij}>$, $|V_{ij}|$ é o número de exemplos de treinamento que satisfazem a condição $<A_i\ Op_i\ V_{ij}>$ e têm o *j-ésimo* valor de G, $|\neg V_i|$ é o número de exemplos de treinamento que não satisfazem a condição $<A_i\ Op_i\ V_{ij}>$, e $|\neg V_{ij}|$ é o número de exemplos de treinamento que não satisfazem $<A_i\ Op_i\ V_{ij}>$ e têm o *j-ésimo* valor de G.

```
/* n = número de genes = número de atributos disponíveis para compor o antecedente da regras
  A i-ésima posição do vetor Info_Gain_Cond[] contem o ganho de informação da i-ésima condição. Esta é
utilizada como a probabilidade a partir da qual a condição é ativada
  A i-ésima posição do vetor Sorted_Cond[] contem a identificação da condição com o i-ésimo menor ganho de
informação*/
INICIO
   Min_N_Cond = 2;      /* Número mínimo de condições */
   PARA i = 1 ATÉ n
      processa Info_Gain_Cond[i];   /* conforme texto */
   FIM PARA
   ordena as  n condições de forma crescente conforme Info_Gain_Cond[i];
   PARA i = 1 ATÉ n
      Sorted_Cond[i] = Identificação da condição com o i-ésimo menor ganho de informação;
   FIM PARA
   Iteration_Id = 1;
   N_Act_Cond = número de condições ativas (com bit ativo = 1) no genoma;
   ENQUANTO (N_Act_Cond > Min_N_Cond) E (Iteration_Id < n)
      Random_N = número gerado aleatoriamente no intervalo 0..1;
      SE Random_N < Info_Gain_Cond[Sorted_Cond[Iteration_Id]]
         ENTÃO condição que Id é Sorted_Cond[Iteration_Id] é ativado (ou seja, presente na regra)
         CASOCONTRÁRIO condição Sorted_Cond[Iteration_Id]
               não é ativada (ou seja, não presente na regra)
FIM ENQUANTO
FIM
```

FIGURA 3.3 PROCEDIMENTO DE PODA DE REGRA APLICADO AOS INDIVÍDUOS DO AG

O uso deste procedimento de poda de regra combina a natureza estocástica dos AGs com a heurística da teoria da informação para decidir quais condições compõem um

antecedente de regra, a qual é uma heurística bastante utilizada por algoritmos de *Data Mining*. Como resultado da adoção deste procedimento, o AG proposto tende a produzir regras com um número menor de condições e ao mesmo tempo contemplando atributos com alto ganho de informação, cujos valores são estimados como sendo mais relevantes para a predição da classe de um exemplo.

Uma descrição mais detalhada do procedimento de poda de regra pode ser encontrada na Figura 3.3. Conforme pode ser verificado nessa figura, o mecanismo iterativo para remoção das condições a partir da regra é implementado a partir da ordenação crescente das condições da regra pelo ganho de informação. Do ponto de vista do AG, esta é uma ordenação lógica, ao invés de uma ordenação física. Em outras palavras, a condições ordenadas são armazenadas em uma estrutura completamente separada em relação à estrutura interna do genoma dos indivíduos.

3.1.5 Classificando os exemplos do conjunto de teste

Deve ser lembrado que o sistema trata cada nó folha como sendo um pequeno ou grande disjunto, e que o AG induz c regras para cada um dos d pequenos disjuntos, onde c é o número de classes e d é o número de pequenos disjuntos.

Uma vez concluídas todas as d x c execuções do AG, os exemplos do conjunto de teste são classificados da seguinte forma:

a) Para cada exemplo de teste é identificado se o mesmo pertence a um pequeno ou grande disjunto, através da árvore de decisão construída pelo C4.5;

b) Se o exemplo pertencer a um grande disjunto, o mesmo será classificado pela árvore de decisão – ou seja, é prevista a classe correspondente à maioria dos exemplos do respectivo nó folha.

c) Caso contrário – ou seja, o exemplo pertence a um nó folha que representa um pequeno disjunto – o sistema tenta classificá-lo através de uma das c regras descobertas pelo AG para o correspondente pequeno disjunto. Neste ponto existem três resultados possíveis:

   c.1) Existe mais de uma regra cobrindo o exemplo, dado que pode haver sobreposição entre as regras descobertas pelo AG. Neste caso o exemplo é classificado pela regra de melhor qualidade, a qual é medida através da função de *fitness* do AG – conforme expressão 3.1.

c.2) Existe apenas uma única regra cobrindo o exemplo. Neste caso o exemplo é simplesmente classificado por esta regra.

c.3) Não existe nenhuma regra descoberta pelo AG que cubra o exemplo. Neste caso o exemplo é classificado pela regra *default*. A regra *default* representa a predição da classe da maioria dos exemplos pertencentes ao pequeno disjunto. Este também é o critério adotado pelo algoritmo C4.5 para definir a regra *default*.

3.2 UM ALGORITMO GENÉTICO PARA DESCOBRIR REGRAS PARA O CONJUNTO TOTAL DE PEQUENOS DISJUNTOS (AG-GRANDE-NS)

3.2.1 A motivação para Descobrir Regras a Partir do Conjunto Total de Pequenos Disjuntos

Na seção anterior foi descrito em detalhes o AG-Pequeno proposto para descoberta de regras de pequenos disjuntos. A idéia central é executar AG-Pequeno múltiplas vezes, sendo que cada execução descobre regras a partir de cada pequeno disjunto separadamente. O algoritmo híbrido C4.5/AG-Pequeno produziu resultados relativamente bons – em geral melhor precisão preditiva que o C4.5 sozinho – conforme será mostrado no próximo capítulo. Porém, no decorrer da pesquisa detectou-se algumas desvantagens do AG-Pequeno, a saber:

(a) Cada execução do AG-Pequeno tem acesso a um pequeno conjunto de treinamento, consistindo de apenas poucos exemplos pertencentes a um único nó folha da árvore de decisão. Intuitivamente, esta característica dificulta, em alguns casos, a indução de regras de classificação confiáveis;

(b) O algoritmo híbrido C4.5/AG-Pequeno tende a descobrir um número maior de regras, comparado ao C4.5 sozinho. Após a identificação dos pequenos disjuntos através da árvore de decisão, o C4.5 sozinho associa a cada pequeno disjunto uma única regra. Em contraste, para cada pequeno disjunto, o AG-Pequeno (como componente do método híbrido) descobrirá c regras, onde c é o número de classes. Desta forma, o método C4.5/AG-Pequeno tende a descobrir conjuntos de regras mais complexas, se comparado com o C4.5 sozinho; e

(c) Embora cada execução do AG-Pequeno seja relativamente rápida (por acessar um conjunto de treinamento muito pequeno), o número alto de execuções do AG faz com

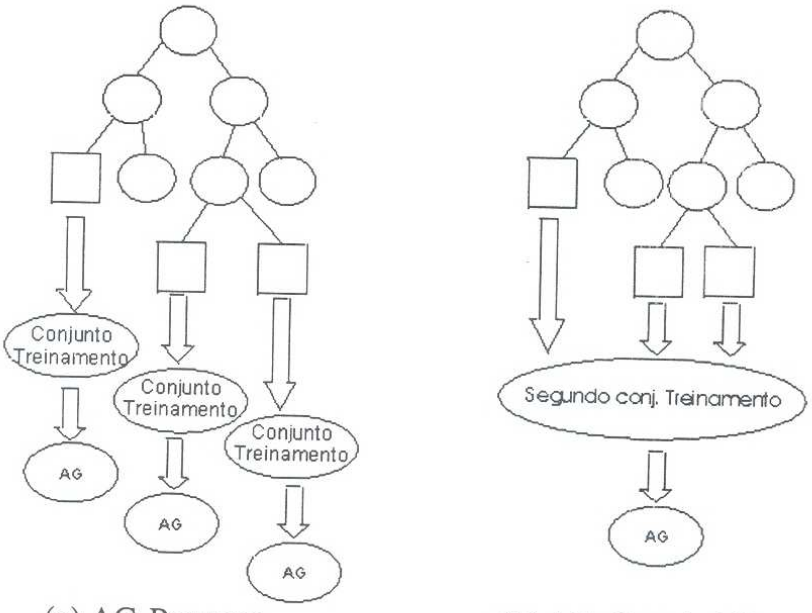
que o sistema como um todo seja consideravelmente mais lento que o C4.5 sozinho, particularmente quando há um grande número de pequenos disjuntos [CAR02b].

A fim de evitar essas desvantagens, este trabalho também propõe um novo AG que pode ser considerado uma significativa extensão do AG-Pequeno. A idéia básica dessa extensão é descrita na próxima seção.

3.2.2 Idéia básica do Algoritmo Genético estendido (AG-Grande-NS)

Esta nova versão proposta do AG é denominada AG-Grande-NS, onde NS indica a adoção de nicho seqüencial (que será descrito a seguir). A principal diferença entre o AG-Pequeno e o AG-Grande-NS é que, neste último, todos os exemplos pertencentes a pequenos disjuntos são agrupados em um único conjunto de treinamento, um conjunto relativamente grande [CAR02a]. Esse conjunto de treinamento, identificado como segundo conjunto de treinamento, é então fornecido como dados de entrada para o AG. Esta característica determina o principal contraste em relação à versão AG-Pequeno.

O conceito de segundo conjunto de treinamento adotado pelo AG-Grande-NS é ilustrado na Figura 3.4(b), onde é possível perceber que todos os pequenos disjuntos são agrupados em um único conjunto de treinamento, relativamente grande. Trata-se de um contraste marcante em relação à abordagem usada pelo AG-Pequeno (descrito na seção 3.1), ilustrada na Figura 3.4(a), onde se verifica claramente que cada pequeno disjunto é usado como um conjunto de treinamento.



(a) AG-Pequeno (b) AG-Grande-NS

FIGURA 3.4 DIFERENÇAS NO CONJUNTO DE TREINAMENTO DOS AGS

O AG-Grande-NS tem a vantagem de propor uma solução que tenta resolver o problema fundamental inerente aos pequenos disjuntos, que é a dificuldade de generalizar corretamente a partir de um conjunto de treinamento muito pequeno. O AG-Grande-NS evita esse problema usando um conjunto de treinamento "grande", o que motivou o uso do termo Grande em sua identificação.

Uma conseqüência imediata desta modificação pode ser observada na tendência a uma sensível redução na cardinalidade do conjunto de regras descobertas. (Essa tendência será evidente na seção de resultados computacionais.)

Conforme pode ser observado na Figura 3.5, no caso do AG-Pequeno o conjunto total de regras descobertas é diretamente dependente do número de nós folhas que representam pequenos disjuntos e do número de classes da base de dados. Ao passo que, no caso do AG-Grande-NS, o número de regras tende a ser bem reduzido. Por exemplo, para a base de dados Connect (Tabela 4.1), uma das bases de dados utilizadas nos experimentos, e considerando uma das definições de pequeno disjunto (seção 4.3), o conjunto de regras descobertas pelo AG-Pequeno é 3 x 227 (3 classes x 227 pequenos disjuntos), ao passo que para a mesma situação o AG-Grande-NS descobriu um conjunto contendo apenas 13 regras.

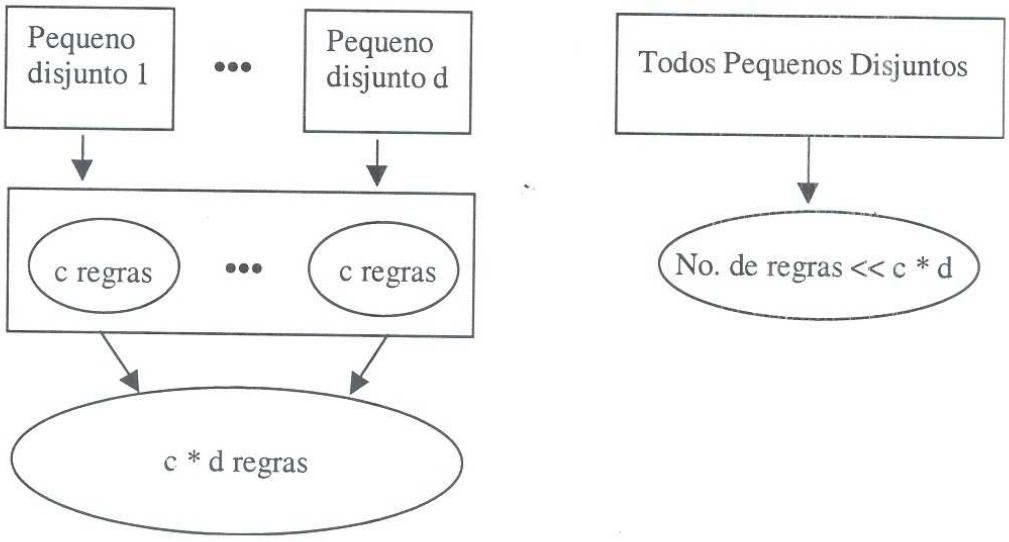


FIGURA 3.5 DIFERENÇA DA CARDINALIDADE DO CONJUNTO DE REGRAS DESCOBERTAS PELOS AGS

Este fato favorece a questão da compreensibilidade do conhecimento descoberto, um resultado desejável em *Data Mining*.

Adicionalmente a esta principal modificação citada anteriormente, o AG-Grande-NS difere do AG-Pequeno em mais quatro aspectos, os quais serão descritos nas próximas seções. De forma geral, estas quatro alterações são derivadas do fato de ser fornecido um

segundo conjunto de treinamento relativamente bem maior (em comparação com um conjunto de treinamento do AG-Pequeno) para a evolução do AG.

3.2.3 Adoção de um método de Nicho Seqüencial

Como resultado do aumento da cardinalidade do conjunto de treinamento, discutido anteriormente, faz-se necessário, na única execução do AG-Grande-NS, descobrir diversas regras que possam cobrir os exemplos de cada classe. (Lembrando que esta preocupação não estava presente na abordagem descrita na seção 3.1, dado que naquela abordagem era assumido que uma execução do AG-Pequeno deveria descobrir uma única regra para cada classe.) Portanto, no AG-Grande-NS é fundamental o uso de algum tipo de método de *niching*, de forma a forçar a diversidade da população e evitar a sua convergência para uma única regra. Neste trabalho foi adotada uma variação do método nicho seqüencial [BEA93]. Este método foi selecionado por duas razões. Primeira, por sua simplicidade. Segunda, e mais importante, é que este método não requer a especificação de parâmetros adicionais para a sua execução, tal como ocorre em métodos de *niching* bem conhecidos como *fitness sharing* [GOL97] e *crowding* [MAH95]. Em particular, o parâmetro σ_{share} em *fitness sharing* é difícil de ser ajustado, e esse problema é evitado pelo nicho seqüencial.

O pseudocódigo da variante de nicho seqüencial adotado neste trabalho é mostrado, em alto nível de abstração, na Figura 3.6. Ele começa pela inicialização do conjunto de regras descobertas (identificado como ConjRegras) com o conjunto vazio, em seguida com a criação do segundo conjunto de treinamento (identificado como ConjTreinamento2), como explicado anteriormente. A partir deste ponto, executa-se de forma iterativa o seguinte *loop*. Primeiro, executa-se o AG, usando ConjTreinamento2 como dados de treinamento. A melhor regra encontrada pelo AG (ou seja, o melhor indivíduo da última geração) é adicionada ao ConjRegras. A partir deste momento os exemplos corretamente cobertos pela regra são removidos do ConjTreinamento2. Desta forma, na próxima iteração do *loop* ENQUANTO o ConjTreinamento2 terá uma cardinalidade menor. Um exemplo é "corretamente coberto" pela regra se os valores dos atributos previsores do exemplo satisfizerem todas as condições no antecedente da regra e o exemplo pertencer à mesma classe prevista pela regra. Este processo é iterativamente executado enquanto o número de exemplos no ConjTreinamento2 for maior que cinco. Desta forma não serão descobertas regras que cubram menos que cinco exemplos, evitando a geração de regras que possivelmente estariam ajustadas demais (*overfitted*) aos

dados. (Assume-se que quando a cardinalidade do ConjTreinamento2 for menor ou igual a cinco não existem exemplos suficientes para permitir a descoberta de regras de classificação confiáveis.)

Os cinco ou menos exemplos não cobertos por nenhuma regra serão classificados pela regra *default*, a qual prediz a classe da maioria destes exemplos. Cabe ressaltar que não está sendo afirmado que cinco seja um valor ótimo para esse *threshold*. Em todo caso, intuitivamente, tendo em vista tratar-se de uma valor pequeno, pequenas variações nesse valor não têm um grande impacto na *performance* do algoritmo. É importante destacar que este número *threshold* define o número máximo de exemplos não cobertos pelo conjunto completo de regras descobertas pelo AG. Em contraste, um número *threshold* análogo, normalmente usado em algoritmos que constróem árvores de decisão, atuando como critério de parada para a expansão da árvore em vários nós folha, intuitivamente tem um impacto significativamente maior na *performance* do algoritmo de árvore de decisão.

```
INICIO
/* ConjTreinamento2 - contém todos os exemplos pertencentes a todos os pequenos disjuntos */
ConjRegras = ∅;
constroi ConjTreinamento2;
ENQUANTO cardinalidade(ConjTreinamento2) > 5
   executa GA;
   adiciona a melhor regra descoberta pelo AG ao ConjRegras;
   remove do ConjTreinamento2 os exemplos corretamente cobertos pela melhor regra;
FIM-ENQUANTO
FIM-INICIO
```

FIGURA 3.6 AG COM NICHO SEQÜENCIAL PARA DESCOBERTA DE REGRAS DE PEQUENOS DISJUNTOS

É importante salientar que o método de nicho seqüencial adotado neste trabalho é uma variação do método proposto por [BEA93]. Este último requer a especificação de um parâmetro, associado a uma métrica de distância, a fim de modificar a paisagem de *fitness* (*fitness landscape*) de acordo com a localização das soluções encontradas nas iterações anteriores. Para a implementação deste parâmetro, o autor usou a distância Euclidiana.

Em contraste, o método de nicho seqüencial adotado neste trabalho não necessita deste tipo de parâmetro. Para evitar que um mesmo espaço de busca seja explorado várias vezes, os exemplos que são corretamente cobertos pela regra descoberta são removidos do conjunto de treinamento. Desta forma, a natureza da paisagem da *fitness* é automaticamente atualizada a partir das regras descobertas ao longo das diversas iterações do método de nicho seqüencial. A variante de nicho seqüencial adotada neste trabalho é essencialmente

equivalente à idéia de "separar-para-conquistar" encontrada em alguns algoritmos de indução de regras.

3.2.4 Modificação do método usado para determinar o conseqüente da regra

Cada execução do AG-Grande-NS descobre uma única regra, e o conseqüente da regra (a classe prevista pela regra) não está codificado no genoma da regra, da mesma forma que no caso do AG-Pequeno descrito na seção 3.1. Entretanto, diferentemente do AG-Pequeno, no AG-Grande-NS o conseqüente de cada regra não é fixado antecipadamente para todas as regras (indivíduos) na população. O conseqüente de cada regra é dinamicamente determinado em função do antecedente da regra. Mais precisamente, o conseqüente da regra corresponde à classe mais freqüente no conjunto de exemplos cobertos pelo antecedente da regra.

3.2.5 Uma nova medida heurística para podar as regras

O AG-Grande-NS proposto neste trabalho adota uma nova medida heurística para a poda de regra. Esta medida é baseada no uso da árvore de decisão construída pelo C4.5 para computar a precisão preditiva (taxa de acerto) de cada atributo previsor, conforme a correspondente precisão obtida através dos percursos na árvore de decisão onde o atributo esteja presente. O atributo de maior precisão preditiva terá menor probabilidade de ter a sua correspondente condição removida da regra. A Figura 3.7 ilustra como é obtida a precisão preditiva do atributo em relação à sua ocorrência na árvore de decisão.

O procedimento para computar a taxa de acerto de cada atributo é mostrado na Figura 3.7. Para cada atributo A_i, o algoritmo verifica todos os percursos da árvore de decisão construída pelo C4.5 com o objetivo de determinar se A_i ocorre no percurso. (O termo percurso é usado aqui para referenciar cada caminho completo desde o nó raiz até o nó folha que representa um pequeno disjunto na árvore.) Para cada percurso p no qual A_i ocorre, o algoritmo computa dois contadores, identificados como número de exemplos classificados pela regra associada ao percurso p, denominado $\#Classif(A_i,p)$, e o número de exemplos corretamente classificados pela regra associada ao percurso p, denominado $\#CorrClassif(A_i,p)$.

A taxa de acerto do atributo A_i (denominada $Acc(A_i)$) sobre todos os percursos nos quais A_i ocorre é obtida através da fórmula:

$$Acc(A_i) = (\sum_{p=1}^{Zi} \#CorrClassif(A_i,p)) / (\sum_{p=1}^{Zi} \#Classif(A_i,p)) \quad (3.5)$$

onde Z_i é o número de percursos da árvore de decisão onde A_i ocorre.

INICIO
Num_Atribtos_não_Usados = 0;
PARA cada atributo A_i, i=1,...,m
SE atributo A_i **ocorre em pelo menos um caminho** de pequeno disjunto da árvore
ENTAO processa a taxa de acerto de A_i, identificada $Acc(A_i)$ (veja texto);
CASO-CONTRARIO incrementa *Num_Atributos_não_Usados* com 1;
FIM-IF
FIM-PARA
Min_Acc = a menor taxa de acerto entre todos atributo que ocorrem em pelo menos em um caminho de pequeno disjunto da árvore;
PARA cada atributo A_i, i=1,...,m, que **não ocorre em pelo menos em um caminho** de pequeno disjunto da árvore;
$Acc(A_i)$ = *Min_Acc / Num_Atributos_não_Usados*;
FIM-PARA

$$Total\_Acc = \sum_{i=1}^{m} Acc(A_i) ;$$

PARA cada atributo A_i, i=1,...,m
Processe a taxa de acerto normalizada de A_i, identificado como $Norm\_Acc(A_i)$, pela fórmula:
$Norm\_Acc(A_i) = Acc(A_i) / Total\_Acc$;
FIM-PARA
FIM-INICIO

FIGURA 3.7 PROCESSAMENTO DA TAXA DE ACERTO DE CADA ATRIBUTO, PARA O PROCEDIMENTO DE PODA DA REGRA

É importante observar que a expressão 3.5 é usada apenas para atributos que ocorrem pelo menos uma vez em algum percurso de pequeno disjunto da árvore. Todos os atributos que não ocorrem em nenhum percurso de pequeno disjunto da árvore, como por exemplo o atributo A_6 (Figura 3.8), têm o seu valor $Acc(A_i)$ determinado pela expressão:

$$Acc(A_i) = Min\_Acc / Num\_Atributos\_não\_Usado \quad (3.6)$$

onde *Min_Acc* e *Num_Atributos_não_Usado* são calculados conforme procedimento mostrado na Figura 3.7.

É importante salientar que essa heurística de atribuir uma taxa de acerto aos atributos que não ocorrem em nenhum percurso da árvore (conjunto A), por mais que essa taxa seja bem inferior às taxas dos atributos que ocorrem em pelo menos um percurso de pequeno disjunto (conjunto B), tem como objetivo permitir que durante a evolução das regras

existam genes ativos contemplando atributos do conjunto A, apesar da baixa probabilidade destes em relação aos atributos do conjunto B.

Finalmente, o valor de $Acc(A_i)$ para todos os atributos A_i, i=1,...,m, é normalizado pela divisão de seu valor pelo *Total_Acc*, o qual é determinado conforme mostrado na Figura 3.7.

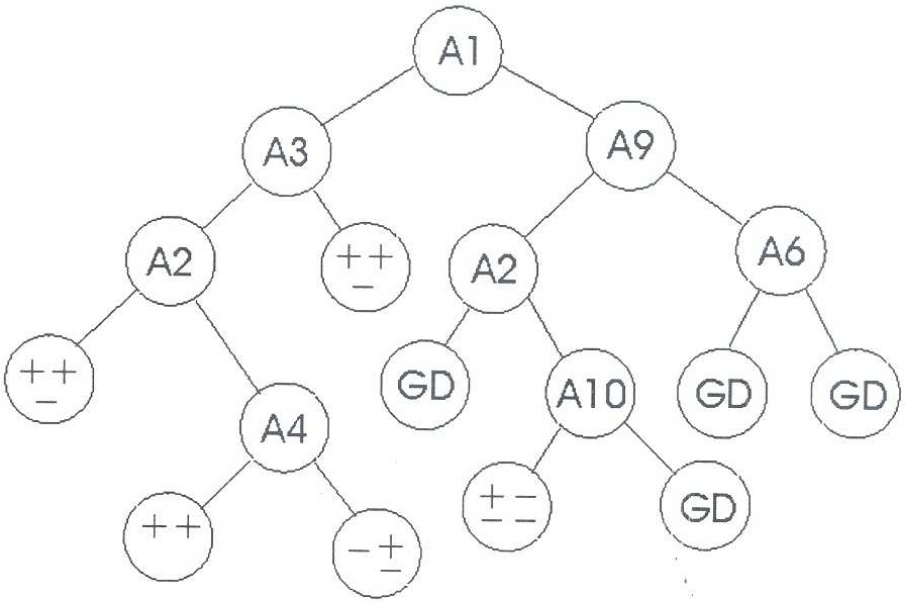


FIGURA 3.8 TAXA DE ACERTO DO ATRIBUTO EM RELAÇÃO À SUA OCORRÊNCIA NA ÁRVORE DE DECISÃO.

Conforme pode ser observado no exemplo da Figura 3.8, a árvore de decisão possui nove nós folhas, dos quais cinco representam pequenos disjuntos (para cada um dos quais é mostrada a distribuição das classes) e quatro representam grandes disjuntos (GD). O cálculo da precisão dos atributos contempla apenas aqueles cinco nós folhas que representam pequenos disjuntos. Por exemplo, o atributo A4 tem uma precisão de 0.8, ou seja, quatro acertos sobre cinco (total de exemplos cobertos pelo atributo). Já o atributo A1 tem uma precisão de 0.73, ou seja 11 acertos sobre o total de 15 exemplos cobertos. Vale lembrar que é considerado acerto o número de exemplos que pertencem à classe da maioria em cada nó folha, ou seja 11 acertos é resultado da somatória (2 "+", 2 "+", 2 "+", 2 "-" e 3 "-").

Uma vez normalizado o valor da taxa de acerto para cada atributo A_i, denominado $Norm\_Acc(A_i)$, este é usado diretamente como uma medida heurística para a poda das regras. A idéia básica é a mesma que a idéia usada no procedimento de poda descrito na seção 3.1. Naquela seção, onde a medida heurística era o ganho de informação, foi mencionado que a condição com maior ganho de informação na regra tinha a menor probabilidade de ser removida. Na versão AG-Grande-NS, o ganho de informação foi substituído pelo $Norm\_Acc(A_i)$, o valor normalizado da taxa de acerto do atributo incluído na condição da

regra. Desta forma, um atributo com o maior valor de $Norm\_Acc(A_i)$ resulta em uma menor probabilidade de que a *i-ésima* condição da regra seja removida. O procedimento restante no processo de poda descrito na seção 3.1 se mantém essencialmente inalterado.

Note que a medida heurística baseada na taxa de acerto para a poda das regras, proposta como parte do AG-Grande-NS, efetivamente usa informação extraída da árvore construída pelo C4.5. Desta forma, ela pode ser considerada como um tipo de medida baseada em hipótese (*hypothesis-driven*), uma vez que ela é baseada na hipótese (neste caso, uma árvore de decisão) previamente construída pelo algoritmo de *Data Mining*.

Em contraste, a medida heurística baseada no ganho de informação, proposta como parte do AG-Pequeno, não usa este tipo de informação. Ela é uma medida que é obtida diretamente através de um conjunto de treinamento, independente de qualquer algoritmo de *Data Mining*. Sendo assim, ela pode ser considerada um tipo de medida baseada em dados (*data-driven*).

3.2.6 Possibilidade de todos os atributos previsores participarem da regra

Deve ser lembrado que um genoma no AG-Pequeno (descrito na seção 3.1) contém apenas os atributos que não são utilizados para rotular os níveis ancestrais de nó folha que for identificado como um pequeno disjunto. Esta abordagem faz sentido tendo em vista que o AG-Pequeno usa como conjunto de treinamento apenas os exemplos pertencentes a um único nó folha. Em geral os atributos nos nós ancestrais ao nó folha em questão não seriam úteis para distinguir as classes dos respectivos exemplos, uma vez que todos estes exemplos têm os mesmos (ou similares) valores para aqueles atributos.

Entretanto, a situação é diferente no caso do AG-Grande-NS. Neste algoritmo, o conjunto de treinamento para o AG consiste de todos os exemplos pertencentes a todos os nós folhas que foram identificados como pequenos disjuntos; ou seja, todos estes exemplos são efetivamente combinados em um único conjunto de treinamento. Desta forma, a noção anteriormente descrita de atributos nos nós ancestrais de um único nó folha deixa de ter sentido. Sendo assim, no AG-Grande-NS o genoma contém m genes, onde m é o número de atributos da base de dados a ser minerada. Concluindo, todos os atributos podem ocorrer na regra representada por um único indivíduo, ou seja, na teoria, uma regra pode conter até m condições no seu antecedente. Claro que, na prática, o número de condições em uma regra deverá ser muito menor que m, dado o uso do operador de poda apresentado anteriormente.

4 RESULTADOS COMPUTACIONAIS

Neste capítulo serão apresentados os resultados obtidos através dos experimentos da seguinte forma. Na seção 4.1 são descritas as bases de dados utilizadas nos experimentos, bem como a metodologia de avaliação. Na seção 4.2 são descritos os classificadores avaliados nos experimentos. Na seção 4.3 são especificados o critério de definição de pequeno disjunto e os parâmetros dos algoritmos. Na seção 4.4 a quantidade total de exemplos em pequenos disjuntos. Na seção 4.5 são apresentados os resultados referentes à taxa de acerto e na seção 4.6 os resultados referentes à simplicidade das regras descobertas. Na seção 4.7 são feitos alguns comentários sobre eficiência computacional. Finalmente, na seção 4.8 são descritos alguns experimentos que auxiliaram a definir a heurística de poda para o algoritmo AG-Grande-NS.

4.1 BASES DE DADOS E METODOLOGIA DE AVALIAÇÃO

Os classificadores híbridos C4.5/AG-Pequeno e C4.5/AG-Grande-NS, propostos no Capítulo 3, foram avaliados em 22 bases de dados do mundo real que estão sumarizadas na Tabela 4.1.

Doze destas bases são de domínio público, obtidas do repositório de bases de dados da UCI (University of California at Irvine): Adult, Connect, Crx, Covertype, Hepatitis, Letter, Nursery, Pendigitis, Segmentation, Splice, House-votes e Wave. Estas bases de dados estão disponíveis no *web site* http://www.ics.uci.edu/~mlearn/MLRepository.html. As outras dez bases são conjuntos de dados derivados de uma base de dados do CNPq, cujos detalhes são confidenciais [ROM02a], [ROM02b]. Esta base contém dados sobre a produção científica de pesquisadores. Todos os dados minerados foram anônimos, ou seja, todos os atributos identificadores de pesquisadores, como nome e CPF, foram removidos. Foram identificados cinco atributos meta possíveis, ou seja, cinco atributos que o usuário desejava predizer, baseado nos valores dos atributos previsores para um determinado exemplo (pesquisador). Estes atributos meta envolvem informação sobre o número de publicações dos pesquisadores, sendo cada atributo meta relacionado a um tipo específico de publicação. Para cada um destes cinco atributos meta (base do CNPq), foram extraídos dois distintos conjuntos de dados, sendo que estes dois conjuntos de dados diferem nos atributos previsores relacionados. Estes conjuntos de dados estão identificados como CD-1,CD-2,...,CD10 na Tabela 4.1.

TABELA 4.1 PRINCIPAIS CARACTERÍSTICAS DAS BASES DE DADOS UTILIZADAS NOS EXPERIMENTOS

Base de dados	No. de exemplos	No. de atributos	No. de classes
Connect	67557	42	3
Adult	45222	14	2
Crx	690	15	2
Hepatitis	155	19	2
House-votes	506	16	2
Segmentation	2310	19	7
Wave	5000	21	3
Splice	3190	60	3
Covertype	8300	54	7
Letter	20000	16	26
Nursery	12960	8	5
Pendigits	10992	16	9
CD-1	5690	23	3
CD-2	5690	23	3
CD-3	5690	23	3
CD-4	5690	23	2
CD-5	5690	23	2
CD-6	5894	22	3
CD-7	5894	22	3
CD-8	5894	22	3
CD-9	5894	22	2
CD-10	5894	22	2

Para a realização dos experimentos relatados neste trabalho foi adotado o seguinte critério:

- para as bases de dados contendo um número de exemplos maior ou igual a 20.000 exemplos foi adotado um único conjunto de treinamento e um conjunto de teste.
- para as bases de dados contendo um número inferior a 20.000 exemplos, foi adotada a validação cruzada estratificada com fator 10, que é um procedimento amplamente utilizado em *Data Mining* [SCH93b], [WIT99].

Assim sendo, as bases para as quais foi adotado um único conjunto de treinamento e de teste foram Adult, Connect e Letter. No caso da base Adult foi usada a divisão pré-definida em base de treinamento e de teste, já disponível no repositório de bases de dados da UCI. Para as bases Connect e Letter, foi feito um particionamento aleatório em conjunto de treinamento e de teste. Uma questão que surge ao se segmentar uma base em dados de treinamento e dados de teste é quantos exemplos devem compor cada uma delas. Tradicionalmente é usada uma porcentagem fixa para o conjunto de treinamento e para o de teste. Estas porcentagens são aproximadamente de 2/3 e 1/3 para treinamento e teste

respectivamente [WEI91]. Respeitando esta orientação, para a realização dos experimentos foi adotada a proporção 70% para a base de treinamento e 30% para a base de teste. Mais especificamente, os conjuntos de treinamento e de teste para a base Connect são compostos de 47290 e de 20267 exemplos, respectivamente. Já para a base Letter os conjuntos de treinamento e de teste são compostos de 14000 e de 6000 exemplos, respectivamente. Cabe ressaltar que, embora a base Letter não seja tão grande quanto as base Connect e Adult, o procedimento de usar uma única partição em treinamento e teste para a base Letter também foi adotado no projeto Statlog [HOR94], um projeto bastante conhecido na literatura. No caso específico dessas três bases de dados, uma única partição de conjunto de treinamento/teste é aceitável, tendo em vista serem três bases relativamente grandes.

Para as demais 19 bases de dados, uma vez que elas não são tão grandes, foi adotada a validação cruzada com o objetivo de tornar os resultados mais confiáveis. Em outras palavras, as bases de dados foram aleatoriamente divididas em dez partições, e cada algoritmo foi executado dez vezes. A freqüência das classes em cada uma das 10 partições foi mantida equivalente à freqüência na base original (validação cruzada estratificada). Em cada uma das execuções de um algoritmo, uma das dez partições foi usada como conjunto de teste e todas as outras nove partições compuseram o conjunto de treinamento. Os resultados relatados para essas 19 bases representam a média sobre as dez execuções [FEE99]. Segundo [WEI91], após a realização de testes com vários valores para o fator de validação cruzada, o valor que resultou como sendo o mais adequado foi 10. O fator 10 de validação cruzada também é considerado o mais usado na prática segundo [WIT99].

Vale ressaltar que exemplos com valores ausentes (*missing values*) foram removidos das bases de dados [WEI98].

4.2 CLASSIFICADORES AVALIADOS NOS EXPERIMENTOS

Para a realização dos experimentos foi utilizado o algoritmo C4.5 [QUI93] como sendo o componente de árvore de decisão dos métodos híbridos identificados como C4.5/AG-Pequeno e C4.5/AG-Grande-NS. Os resultados obtidos a partir destas duas versões do método híbrido foram comparados com os resultados de cinco algoritmos, conforme a seguir:

a) três versões do C4.5 sozinho;
b) o método híbrido proposto por Ting [TIN94] – árvore de decisão / IBL (C4.5/IB1), o qual constrói classificadores para tratar o problema do pequeno disjunto;

c) um Algoritmo Genético sozinho (AG-Sozinho) que descobre regras tanto para pequenos disjuntos quanto para grandes disjuntos.

Esses cinco algoritmos são descritos a seguir, iniciando com as três versões do C4.5 sozinho. A primeira versão consiste da simples execução do C4.5, com valores *default* para seus parâmetros, e considerando como resultado daquela execução a árvore podada. Maiores detalhes sobre este processo de poda podem ser encontrados em [QUI93].

A segunda versão consiste na mesma execução do C4.5 (também adotando os valores de parâmetros *default*), porém considerando agora como resultado a árvore não-podada. É sabido que, em geral, mas não sempre, os resultados do C4.5 com poda são melhores que os resultados do C4.5 sem poda. Entretanto, no contexto deste trabalho existe a motivação para avaliar os resultados do C4.5 sem poda. Este trabalho tem como objetivo descobrir regras para pequenos disjuntos, as quais tendem a ser regras mais específicas. Naturalmente, uma árvore não podada contém regras mais específicas do que uma árvore podada. Por outro lado, existe o risco do C4.5 sem poda se ajustar demais aos dados (*overfitting*), o que pode acarretar regras específicas demais para os dados de treinamento, as quais não representariam generalizações válidas para os dados de teste. De qualquer forma é interessante avaliar o C4.5 sem poda como uma solução alternativa para o problema do pequeno disjunto, uma vez que se trata de uma solução simples e que também serve como padrão de comparação para avaliar outras soluções propostas para o problema de pequenos disjuntos. Em ambas as versões (com poda e sem poda) a árvore de decisão construída é usada para classificar tanto os exemplos de grandes disjuntos quanto os de pequenos disjuntos.

A terceira versão consiste de uma "dupla execução" do C4.5, identificada como C4.5 duplo. Trata-se de uma nova forma de utilizar o C4.5 para tratar do problema do pequeno disjunto, que tem como idéia a construção de um classificador a partir da execução do C4.5 duas vezes. A primeira execução considera todos os exemplos do conjunto original de treinamento para a construção da primeira árvore de decisão. Uma vez que todos os exemplos pertencentes a pequenos disjuntos tenham sido identificados por esta primeira árvore de decisão, o sistema agrupa todos estes exemplos em um único conjunto de treinamento, criando um segundo conjunto de treinamento (um subconjunto do conjunto original), como descrito anteriormente para o AG-Grande-NS (Figura 3.5(b)). Uma vez obtido esse segundo conjunto de treinamento, uma segunda execução do C4.5 é realizada, só que agora tendo como entrada apenas esse segundo conjunto. Note que esse segundo conjunto é o mesmo utilizado pelo AG-Grande-NS. Para classificar um novo exemplo, as regras

descobertas pelas duas execuções do C4.5 são utilizadas da seguinte forma: Primeiramente, o sistema verifica se o novo exemplo pertencente a um grande disjunto da primeira árvore de decisão. Em caso afirmativo, a classe predita pelo correspondente nó folha é atribuída ao novo exemplo. Caso contrário (o novo exemplo pertence a um pequeno disjunto na primeira árvore de decisão), o exemplo é classificado pela segunda árvore de decisão. A motivação para este uso mais elaborado do C4.5 consiste em uma tentativa de criar um algoritmo simples que seja mais efetivo no tratamento de pequenos disjuntos.

O método híbrido proposto por [TIN94], denominado C4.5/IB1, funciona do seguinte modo. Inicialmente o C4.5 é executado com valores *default* para seus parâmetros. A seguir cada nó folha da árvore podada produzida pelo C4.5 é considerado como um pequeno ou grande disjunto, utilizando o mesmo critério adotado neste trabalho para definição de pequeno disjunto (seção 4.3). Exemplos pertencentes a grandes disjuntos são classificados pelo C4.5, enquanto exemplos pertencentes a pequenos disjuntos são classificados por um algoritmo de aprendizado baseado em instâncias denominado IB1[AHA91]. O algoritmo IB1 é executado uma vez para cada pequeno disjunto. Cabe ressaltar que IB1 é um algoritmo simples. Ele não utiliza pesos de atributos e nem pesos de exemplos, ao contrário de algoritmos de aprendizagem baseado em instâncias mais sofisticado [AHA98]. Além disso, ele usa apenas um vizinho mais próximo para classificar o novo exemplo de teste.

Assim, em um alto nível de abstração, a idéia básica desse método híbrido C4.5/IB1 é semelhante à idéia básica do método híbrido C4.5/AG-Pequeno. O híbrido C4.5/IB1 pode ser entendido substituindo-se, na Figura 3.4, as execuções do AG-Pequeno pelas execuções do IB1. A motivação para o método híbrido C4.5/IB1, conforme mencionado por [TIN94], é que algoritmos baseados em instâncias têm um *bias* de especificidade (classificando um novo exemplo dos dados de teste com base em um pequeno número de exemplos semelhantes nos dados de treinamento), o qual, intuitivamente, parece ser um *bias* adequado para tratar o problema de pequenos disjuntos. Porém, cabe ressaltar que o IB1 tem a desvantagem de não descobrir regras da forma se ... então, o que prejudica a compreensibilidade do conhecimento descoberto. Uma discussão mais detalhada do trabalho de [TIN94] é apresentada no Capítulo 5.

O algoritmo denominado AG-Sozinho consiste essencialmente na execução do AG-Grande-NS utilizando, como dados de entrada, a base de treinamento inteira. Assim sendo, o AG-Sozinho é usado para classificar todos os exemplos, sem haver distinção entre exemplos

de pequenos ou grande disjuntos. Isso é análogo à execução do C4.5 sozinho, onde também é ignorada a distinção entre exemplos de pequenos e grandes disjuntos.

Portanto, o uso do C4.5 sozinho e do AG-Sozinho nos experimentos é interessante, já que esses dois métodos representam dois tipos de algoritmos "opostos", e o método híbrido C4.5/AG-Grande-NS proposto neste trabalho pode ser visto como uma solução intermediária entre aqueles dois opostos.

Na implementação do AG-Sozinho, foi feita apenas uma alteração em relação ao AG-Grande-NS, com relação à heurística de poda baseada na taxa de acerto. Esta alteração se deve ao fato de que no caso do AG-Sozinho, não há distinção entre pequenos e grande disjuntos. Portanto, foi necessário alterar o cálculo dos valores das variáveis número de exemplos cobertos (*#Classif*(A_i,p)) e número de exemplos corretamente classificados (*#CorrClassif*(A_i,p)) pelos percursos p's (Figura 3.7). No caso da versão AG-Grande-NS, p's representam os percursos referentes a pequenos disjuntos. Já no caso do AG-Sozinho, p representa todo e qualquer percurso da árvore. (Note que o AG-Sozinho utiliza informação da árvore de construída pelo C4.5 na heurística de poda de regras apenas. O AG-Sozinho não utiliza a árvore gerada pelo C4.5 para classificação de nenhum exemplo.) As demais características do AG-Sozinho são idênticas às do AG-Grande-NS.

4.2.1 Implementação dos experimentos

Para a realização dos experimentos foi utilizado o algoritmo C4.5, estando seu fonte disponível em [QUI93]. A árvore gerada pelo C4.5 foi armazenada em um arquivo no formato texto, o qual foi posteriormente fornecido como entrada para um programa implementado em linguagem "C" padrão, especificamente construído para converter aquela árvore em um conjunto de regras.

Para o desenvolvimento do componente AG dos algoritmos C4.5/AG-Pequeno e C4.5/AG-Grande-NS também foi usada a linguagem "C" padrão. Os algoritmos foram implementados no ambiente *Microsoft Developer Studio Standard Edition* 4.0.

Para a realização dos experimentos com o algoritmo híbrido proposto por Ting [TIN94] entrou-se em contato com o autor para verificar a possibilidade do mesmo disponibilizar o *software* usado por ele na execução dos experimentos realizados em seu trabalho. Ele respondeu que não dispunha mais do mesmo, quando então ele recomendou que fosse utilizado o algoritmo C4.5 para o componente árvore de decisão e qualquer ferramenta

disponível na Internet que implementasse a versão IB1. Além disso, o uso do C4.5 é plenamente justificado a fim de realizar uma comparação justa com o método híbrido proposto nesta tese, o qual também utilizou o C4.5. Desta forma, foi adotado o C4.5 [QUI93] como componente árvore de decisão. Para implementar o algoritmo IB1, foi usada a ferramenta WEKA [WIT99] a partir do componente IBL. A ferramenta WEKA foi obtida através do *web site* http://www.cs.waikato.ac.nz/ml/weka.

4.3 DEFINIÇÃO DE PEQUENO DISJUNTO E PARÂMETROS DOS ALGORITMOS

Como descrito no Capítulo 3, os sistemas híbridos árvore de decisão/AG usam o AG para descobrir regras para classificar apenas os exemplos dos pequeno disjunto, sendo a árvore de decisão a responsável pela classificação dos exemplos dos grandes disjuntos. Intuitivamente, a *performance* do sistema proposto é dependente do critério usado para definição de pequeno disjunto.

Para os experimentos realizados neste trabalho foi adotado um critério simples e comum na prática para a definição de pequeno disjunto, baseado em um número máximo de exemplos cobertos pelo disjunto. A definição é: "Um nó folha da árvore de decisão é considerado um pequeno disjunto se e somente se o número de exemplos pertencentes àquele nó for menor ou igual a um número previamente fixado S". Como forma de avaliar a sensibilidade das duas versões do método proposto em relação ao tamanho do pequeno disjunto, são relatados os resultados para experimentos com quatro valores distintos para o parâmetro S, a saber, $S = 3$, $S = 5$, $S = 10$ e $S = 15$.

Estes valores para S foram obtidos a partir das seguintes considerações:

- no trabalho de Quinlan [QUI91] foram realizados experimentos variando o valor de S entre 1 e 10;
- no trabalho de Danyluk e Provost [DAN93], considerando uma base de treinamento contendo 500 exemplos, a maior concentração de pequenos disjuntos ocorreu para $S = 15$;
- no trabalho de Holte e Porter [HOL89] foi identificado que, dentre todos os erros de classificação, 97 % dos erros ocorrem para o valor de $S = 13$.

A partir da experiência destes autores foi optada por uma definição baseada em um número máximo fixo de exemplos cobertos por um disjunto, bem como os valores 3, 5, 10 e 15. Não foram arbitrados mais valores dentro deste intervalo por dois motivos. Primeiramente, o tempo de processamento para a realização dos experimentos é bastante alto,

dado que para 19 das 22 bases de dados foi adotado a validação cruzada com fator 10 (conforme será mostrado na seção 6.2, o uso de 4 valores diferentes de S resultou em 7720 execuções do algoritmo AG-Grande-NS e 3860 execuções do AG-Pequeno). Segundo, os quatro valores de S citados anteriormente intuitivamente representam uma gama razoável de diferentes tamanhos de pequenos disjuntos.

Cabe ressaltar que um valor de S bem maior que 15 não teria muito sentido, pois perderia o significado de "pequeno" disjunto. Assume-se que em geral o algoritmo C4.5 é adequado para classificar corretamente os exemplos pertencentes aos grandes disjuntos. O AG é acionado para classificar apenas os exemplos pertencentes aos pequenos disjuntos. Se um nó folha da árvore de decisão produzida pelo algoritmo C4.5 tiver muito mais de 15 exemplos, o C4.5 provavelmente teria exemplos suficientes, naquele nó, para realizar uma boa classificação (caso contrário ele provavelmente particionaria este nó, aumentando a ramificação da árvore). É importante salientar que não está sendo feita uma defesa de que estes valores de S são "ótimos". (Um trabalho de otimização do valor de S poderia ser realizado futuramente.)

A primeira vista um valor fixo para S pode parecer inadequado para bases com número de exemplos muito diferenciados. Porém, vale lembrar também que não é uma tarefa trivial definir um valor relativo de S (em função do número de exemplos na base de dados) pelo mesmo motivo. Por exemplo, suponha que fosse especificado S = 1% dos exemplos. Considerando a base Connect e a base Hepatitis, 1% dos exemplos representa 675 exemplos para a Connect e cinco exemplos para a base Hepatitis. Até que ponto um conjunto de 675 exemplos poder ser considerado um "pequeno" disjunto? Por outro lado, apenas cinco exemplos (Hepatitis) constitui um disjunto bastante pequeno.

O trabalho do Ting [TIN94] realizou experimentos considerando uma definição relativa para a definição de S (4%). Porém, as bases de dados utilizadas nos experimentos relatados não apresentavam muita diversidade de tamanho em relação ao número de exemplos, ao contrário do que ocorre nos experimentos relatados neste trabalho. Na verdade o trabalho do Ting ilustra a dificuldade de se utilizar uma definição relativa de pequeno disjunto em bases de tamanho muito diferentes, ao lidar com a base Wave. Essa base tem 5000 exemplos. Se Ting utilizasse a definição de S = 4% (como ele utilizou para as demais bases em seus experimentos), isso implicaria em pequenos disjuntos cobrindo até 200 exemplos, o que novamente parece um número de exemplos alto demais para um pequeno disjunto. Além disso, no caso específico dessa base, praticamente todos os exemplos seriam considerados

como pequenos disjuntos, já que essa base contém um número alto de pequenos disjuntos (conforme será visto na seção 4.4). Para evitar esse problema, Ting trabalhou com um pequeno subconjunto de treinamento da base Wave, contendo apenas 300 exemplos. Assim, um valor relativo de $S = 4\%$ correspondeu a pequenos disjuntos cobrindo até 12 exemplos.

Essa abordagem possibilita utilizar uma definição relativa de S para todas as bases, mas tem a importante desvantagem de utilizar apenas um pequeno subconjunto de treinamento.

Nos experimentos com AG-Pequeno, apenas dois valores de S foram usados, $S = 10$ e $S = 15$, uma vez que é desejável que o valor de S não seja excessivamente pequeno, dado que nesse caso não existiriam exemplos suficientes para treinar o AG-Pequeno adequadamente. Os demais classificadores – C4.5 sem poda, C4.5 com poda, C4.5 duplo, AG-Sozinho, C4.5/IB1 e C4.5/AG-Grande-NS – não têm essa restrição e, portanto, foram executados com $S = 3$, $S = 5$, $S = 10$ e $S = 15$.

Nos experimentos relativos à versão C4.5/AG-Grande-NS, para cada valor de S especificado, foram realizados dez experimentos diferentes, variando a semente aleatória na geração da população inicial de indivíduos do AG. Para a obtenção da taxa de acerto referente a cada valor de S, foi feita uma média aritmética dos resultados sobre estes dez experimentos diferentes. Sendo assim, o número total de experimentos é 100 (validação cruzada com fator 10 * 10 sementes aleatórias distintas) para cada valor de S em cada base de dados, com exceção das bases Adult, Connect e Letter, para as quais não foi necessário utilizar validação cruzada, conforme discutido anteriormente.

Para a versão C4.5/AG-Pequeno também foram realizados dez experimentos variando a semente aleatória. Porém, é importante observar que o número de execuções do C4.5/AG-Pequeno é muito maior do que 100 para cada base de dados. Para cada valor de semente aleatória e cada partição de validação cruzada, AG-Pequeno é executado $c * d$ vezes, onde c é o número de classes e d é o número de pequenos disjuntos.

Para a versão AG-Sozinho foram realizados também dez experimentos variando a semente aleatória em cada partição de validação cruzada.

Cabe lembrar que o algoritmo de árvore de decisão que compõe o método híbrido proposto neste trabalho é o algoritmo C4.5 com valores *default* de seus parâmetros. Para tornar a comparação mais justa, também não foram adotadas medidas de otimização dos valores dos parâmetros dos algoritmos AG-Pequeno, AG-Grande-NS e AG-Sozinho, tais como tamanho da população, número de gerações, probabilidade de mutação e de

cruzamento. Foram adotados valores comuns para estes parâmetros, sugeridos pela literatura. Mas precisamente, para todos os experimentos, em cada execução dos AGs a população é de 200 indivíduos, o número de gerações é 50 e as probabilidades de cruzamento e mutação são 80% e 1%, respectivamente. Apesar de serem valores sugeridos pela literatura não quer dizer que tratem-se de valores ótimos, o que torna justa a comparação com o C4.5 com valores *default*.

Foram realizados vários conjuntos de experimentos, comparando a precisão preditiva e a simplicidade dos classificadores gerados pelo algoritmos híbridos C4.5/AG-Pequeno e C4.5/AG-Grande-NS com relação às três versões diferentes do algoritmo C4.5, ao AG-Sozinho e ao híbrido C4.5/IB1. Conforme mencionado na seção 2.1, tratam-se de dois critérios relevantes para avaliar a qualidade das regras descobertas na tarefa de classificação. Esses resultados são apresentados nas próximas seções.

4.4 OBSERVAÇÕES SOBRE A QUANTIDADE TOTAL DE EXEMPLOS EM PEQUENOS DISJUNTOS E O NÚMERO DE PEQUENOS DISJUNTOS

Nos experimentos relatados nas seções anteriores, deve ser observado que a porcentagem dos exemplos pertencentes aos pequenos disjuntos é representativa na maioria das bases de dados, confirmando os resultados de [WEI00]. A Figura 4.1 mostra a porcentagem de exemplos de treinamento pertencentes aos pequenos disjuntos para os quatro valores de S, $S = 3$, $S = 5$, $S = 10$ e $S = 15$. Essa porcentagem é calculada como o número total de exemplos pertencentes aos nós folha da árvore de decisão identificados como pequenos disjuntos dividido pelo número total de exemplos.

A porcentagem de exemplos de pequenos disjuntos é representativa particularmente quando $S = 15$. Especificamente, na base Wave mais de 50% dos exemplos pertencem a pequenos disjuntos. Além disso, a porcentagem de exemplos de pequenos disjuntos é maior do que 10% em 14 das 22 bases de dados, quando $S = 15$.

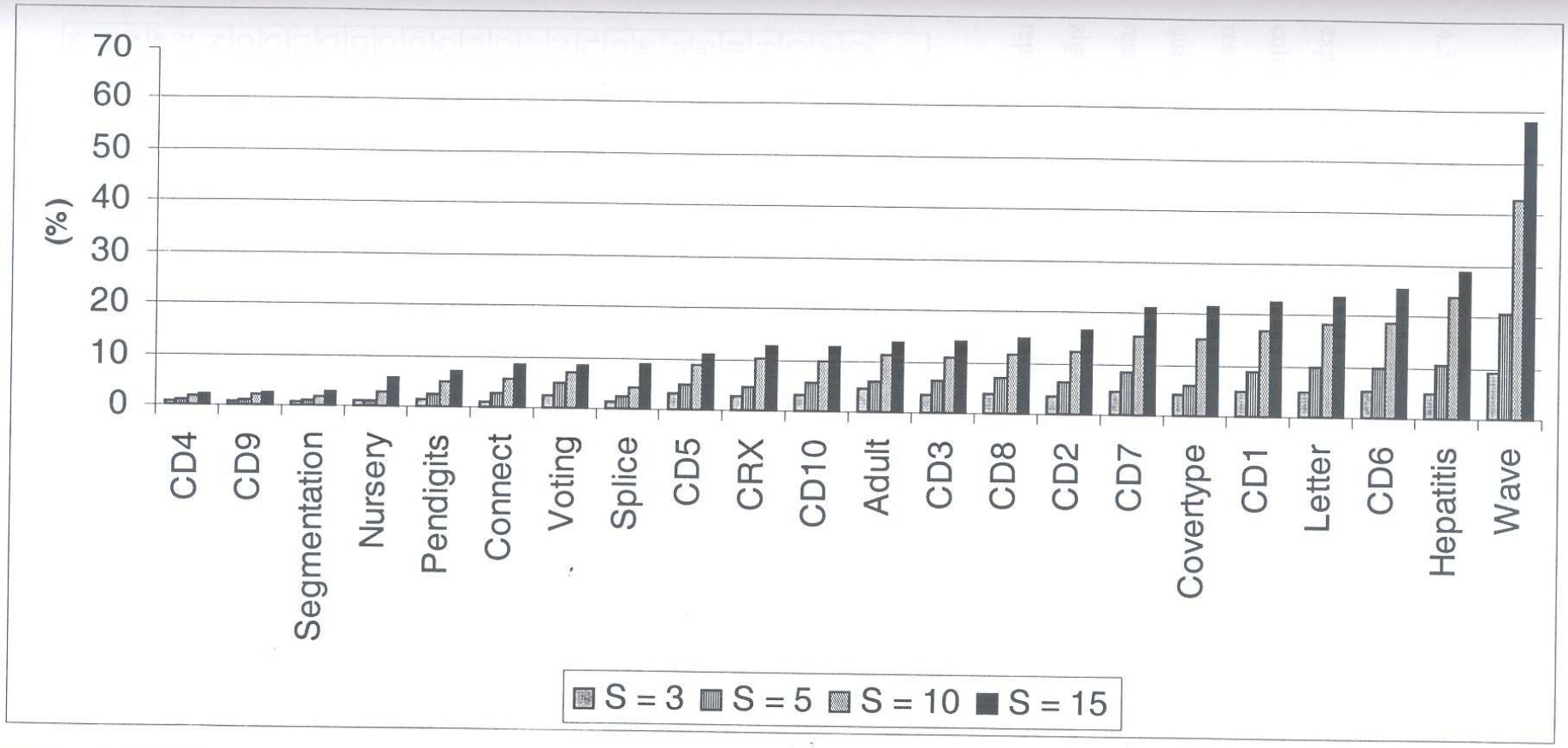


FIGURA 4.1 FREQÜÊNCIA RELATIVA DOS EXEMPLOS DE PEQUENOS DISJUNTOS IDENTIFICADOS NAS BASES DE DADOS UTILIZADAS NOS EXPERIMENTOS DESTE TRABALHO

4.5 RESULTADOS REFERENTES À TAXA DE ACERTO

As Tabelas 4.2 e 4.3 apresentam as taxas de acerto (nos dados de teste) obtidas nos experimentos com as 22 bases de dados, para $S = 3$ e $S = 5$, respectivamente. A primeira coluna identifica as bases de dados, a segunda, a terceira e a quarta coluna mostram os resultados para as três versões do C4.5: com poda, sem poda e duplo, respectivamente. A quinta coluna apresenta os resultados para AG-Sozinho e a sexta coluna apresenta os resultados para o algoritmo híbrido C4.5/IB1. A sétima coluna apresenta os resultados para o algoritmo híbrido C4.5/AG-Grande-NS. Os valores identificados pelo símbolo "±" representam o desvio padrão.

TABELA 4.2 TAXA DE ACERTO (%) PARA $S = 3$

Base Dados	C4.5 com poda	C4.5 sem poda	C4.5 duplo	AG-Sozinho	C4.5/IB1	C4.5/AG-Grande-NS
Connect	72,60 ± 0,5	72,05 ± 0,5	78,06 ± 0,6 +	74,40 ± 0,6 +	**78,13 ± 0,5+**	77,86 ± 0,1 +
Adult	78,62 ± 0,5	77,00 ± 0,5 -	81,19 ± 0,5 +	78,90 ± 0,3	**85,87 ± 0,5+**	85,45 ± 0,1 +
Crx	91,79 ± 2,1	92,45 ± 1,9	92,57 ± 1,2	78,04 ± 1,2 -	92,97 ± 1,5	**93,69 ± 1,2**
Hepatitis	80,78 ± 13,3	77,50 ± 11,3	78,95 ± 6,9	81,28 ± 11,2	**95,84 ± 8,3**	89,25 ± 9,5
House-votes	93,62 ± 3,2	93,50 ± 1,7	97,32 ± 2,4	**97,63 ± 1,6**	96,22 ± 3,1	97,18 ± 2,5
Segmentation	**96,86 ± 1,1**	96,30 ± 0,9	76,62 ± 2,8 -	72,42 ± 4,6 -	81,45 ± 1,2 -	81,46 ± 1,1 -
Wave	75,78 ± 1,9	75,4 ± 2,1	68,18 ± 3,7 -	66,74 ± 4,6 -	83,81 ± 2,0+	**83,86 ± 2,0 +**
Splice	65,68 ± 1,3	66,54 ± 1,3	55,65 ± 6,0 -	60,26 ± 5,0	70,54 ± 8,9	**70,62 ± 8,6**
Covertype	71,61 ± 1,9	70,34 ± 1,9	72,88 ± 14,4	65,40 ± 3,1 -	73,03 ± 13,9	**73,04 ± 1,2**
Letter	**86,4 ± 1,1**	86,30 ± 1,1	83,82 ± 1,0 -	75,80 ± 0,3 -	86,17 ± 1,1	84,26 ± 0,2 -
Nursery	95,4 ± 1,2	**96,40 ± 0,9**	95,55 ± 0,6	82,13 ± 4,4 -	95,48 ± 0,6	95,35 ± 1,2
Pendigits	96,39 ± 0,2	96,36 ± 0,3	97,43 ± 0,3 +	89,01 ± 0,5 -	97,41 ± 0,2+	**97,54 ± 0,3 +**
CD-1	60,71 ± 3,0	58,26 ± 2,8	**62,75 ± 0,6**	61,40 ± 0,3	62,55 ± 3,5	62,53 ± 1,0
CD-2	65,55 ± 1,5	63,22 ± 1,6	**68,95 ± 4,8**	62,95 ± 5,1	68,44 ± 4,7	68,44 ± 2,3
CD-3	75,65 ± 2,4	71,64 ± 1,3 -	80,47 ± 2,1 +	69,24 ± 2,2 -	80,26 ± 1,9+	**80,54 ± 0,9 +**
CD-4	**92,97 ± 0,9**	89,47 ± 0,8 -	92,75 ± 1,4	89,7 ± 2,9	92,72 ± 1,1	92,72 ± 1,1
CD-5	82,7 ± 2,8	78,71 ± 1,9	89,74 ± 2,4 +	72,31 ± 2,2 -	90,15 ± 2,6	**90,17 ± 2,5 +**
CD-6	57,78 ± 2,1	55,36 ± 2,3	59,72 ± 2,4	59,37 ± 3,7	**59,78 ± 2,5**	59,65 ± 1,2
CD-7	65,18 ± 1,0	60,68 ± 1,4 -	**69,94 ± 1,5 +**	65,90 ± 1,6	69,75 ± 1,4+	69,66 ± 0,7 +
CD-8	75,57 ± 1,4	70,30 ± 1,7 -	**80,52 ± 2,3 +**	73,69 ± 1,9	80,45 ± 2,0+	80,41 ± 2,0 +
CD-9	93,00 ± 0,5	89,67 ± 1,4 -	93,72 ± 0,7	87,49 ± 2,9 -	93,86 ± 1,5	**93,87 ± 1,4**
CD-10	82,80 ± 1,7	78,45 ± 2,2 -	85,89 ± 0,5 +	73,89 ± 1,4 -	85,80 ± 1,3+	**85,90 ± 1,2 +**
N. de melhoras significativas		0	8	1	8	9
N. de pioras significativas		7	4	11	1	2

Para cada base de dados, a maior taxa de acerto entre os seis métodos é mostrada em negrito. Nas últimas cinco colunas está indicado, para cada base de dados, se a taxa de

acerto obtida pelo método referenciado na coluna é significativamente diferente (melhor/pior) do que a taxa obtida através do C4.5 com poda, o qual é considerado um método *baseline* para comparação com os outros cinco métodos. Isso permite avaliar o quanto os métodos em questão podem ser considerados boas soluções para o problema do pequeno disjunto. Mais precisamente, os casos onde a taxa de acerto de cada um dos cinco métodos é significativamente melhor (pior) que a taxa de acerto do C4.5 com poda é indicado pelo símbolo "+" ("-"). Uma diferença entre dois métodos é considerada significativa quando não houver sobreposição no intervalo da taxa de acerto correspondente (levando em conta os valores de desvio padrão para os dois métodos).

A penúltima (última) linha das Tabelas 4.2 e 4.3 indica o número de bases de dados onde cada um dos cinco métodos obteve um melhora (piora) significativa na taxa de acerto, em relação ao C4.5 com poda.

Note que as taxas de acerto das versões C4.5 com e sem poda, bem como as do AG-Sozinho, nas Tabelas 4.2, 4.3, 4.4 e 4.5 são exatamente as mesmas, uma vez que estas taxas independem do valor de S.

Analisando os resultados da Tabela 4.2, para $S = 3$, pode-se concluir que dentre os seis métodos os que obtiveram os piores resultados são o C4.5 na sua versão sem poda e o AG-Sozinho. O resultado do C4.5 sem poda chega a ser surpreendente, dado que a árvore não podada poderia ser uma boa alternativa de classificador em bases de dados com um número significativo de pequenos disjuntos, uma vez que a poda da árvore (C4.5 com poda) elimina ramificações que tratam de forma mais adequada os casos raros/exceções. O C4.5 sem poda obteve o melhor resultado em apenas uma das 22 bases e mesmo assim sem auferir uma vantagem significativa.

Ainda sobre a Tabela 4.2, o C4.5 com poda obteve os melhores resultados em três das 22 bases. O C4.5 duplo obteve os melhores resultados em quatro bases. Ao comparar especificamente o C4.5 sem poda com o C4.5 com poda, o C4.5 sem poda obteve piores taxas de acerto, de forma significativa, em sete das 22 bases e melhora significativa em nenhuma. Comparando o C4.5 duplo com o C4.5 com poda, o primeiro foi significativamente melhor em oito e significativamente pior em quatro dentre as 22 bases. O AG-Sozinho obteve apenas uma taxa de acerto significativamente melhor que o C4.5 com poda, porém em 11 bases foi significativamente pior. O C4.5/IB1 obteve a melhor taxa de acerto em quatro bases. Esse algoritmo obteve 8 taxas de acerto significativamente melhores que o C4.5 com poda e apenas uma significativamente pior. O C4.5/AG-Grande-NS obteve as melhores taxas de acerto em

nove das 22 bases. Esse método também obteve taxas de acerto significativamente melhores que o C4.5 com poda em nove bases, e piores, de forma significativa, em apenas duas bases.

Vale destacar que independentemente das análises de qual foi o método que obteve a melhora taxa de acerto e se a melhora (piora) foi significativa em relação ao C4.5 com poda, vários métodos obtiveram uma melhor taxa de acerto em relação ao C4.5 com poda. Por exemplo, o AG-Sozinho obteve taxas de acerto superiores às taxa obtidas pelo C4.5 com poda em sete das 22 bases de dados, apesar de terem ocorrido pioras significativas em 11 bases. Isso significa dizer que, apesar do AG-Sozinho não ser tão competitivo quanto a maioria dos outros métodos, trata-se de um método válido para compor o conjunto de métodos que estão sendo utilizados para avaliar a *performance* dos métodos propostos neste trabalho.

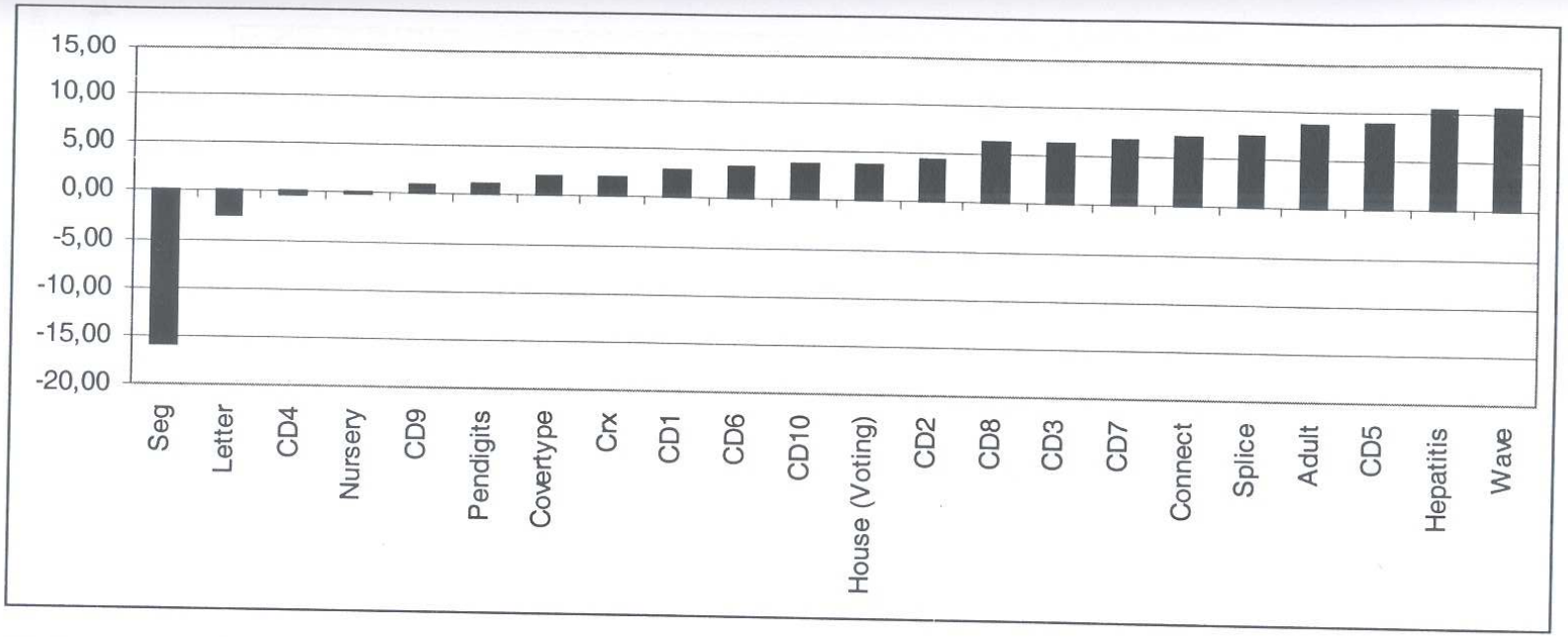


FIGURA 4.2 VARIAÇÃO (%) DA TAXA DE ACERTO DO C4.5/AG-GRANDE-NS EM RELAÇÃO À TAXA DE ACERTO DO C4.5 COM PODA PARA S = 3

Conforme pode ser observado na Figura 4.2, ao serem comparadas as taxas de acerto do C4.5/AG-Grande-NS com as taxas obtidas a partir do C4.5 com poda ($S = 3$), para a grande maioria das bases de dados houve uma melhora na taxa de acerto, chegando a pouco mais de 10% para as bases Hepatitis e Wave, e chegando a mais de 5% em várias outras bases de dados. Apenas na base Segmentation houve uma piora na taxa de acerto de pouco mais de 15%; em nenhuma outra base a piora chegou a 5%.

TABELA 4.3 TAXA DE ACERTO (%) PARA $S = 5$

Base Dados	C4.5 com poda	C4.5 sem poda	C4.5 duplo	AG-Sozinho	C4.5/IB1	C4.5/AG-Grande-NS
Connect	72,60 ± 0,5	72,05 ± 0,5	77,09 ± 0,6 +	74,40 ± 0,6 +	**78,19 ± 0,5 +**	77,85 ± 0,2 +
Adult	78,62 ± 0,5	77,00 ± 0,5 -	79,27 ± 0,5	78,90 ± 0,3	**85,94 ± 0,5 +**	85,50 ± 0,2 +
Crx	91,79 ± 2,1	92,45 ± 1,9	92,03 ± 1,0	78,04 ± 1,2 -	92,54 ± 1,2	**93,06 ± 1,6**
Hepatitis	80,78 ± 13,3	77,50 ± 11,3	75,67 ± 17,1	81,28 ± 11,2	86,53 ± 10,0	**89,48 ± 9,7**
House-votes	93,62 ± 3,2	93,50 ± 1,7	93,54 ± 3,9	**97,63 ± 1,6**	97,04 ± 1,1	97,44 ± 2,9
Segmentation	**96,86 ± 1,1**	96,30 ± 0,9	74,49 ± 3,4 -	72,42 ± 4,6 -	80,22 ± 1,0 -	80,41 ± 1,0 -
Wave	75,78 ± 1,9	75,4 ± 2,1	65,59 ± 4,4 -	66,74 ± 4,6 -	85,33 ± 2,1 +	**85,37 ± 2,4 +**
Splice	65,68 ± 1,3	66,54 ± 1,3	57,45 ± 8,7	60,26 ± 5,0	70,37 ± 8,2	**70,44 ± 7,8**
Covertype	71,61 ± 1,9	70,34 ± 1,9	71,34 ± 14,4	65,40 ± 3,1 -	71,63 ± 14,3	**71,66 ± 1,3**
Letter	86,4 ± 1,1	86,30 ± 1,1	83,62 ± 1,0 -	75,80 ± 0,3 -	**88,15 ± 1,1**	83,28 ± 0,2 -
Nursery	95,4 ± 1,2	96,40 ± 0,9	**96,57 ± 0,7**	82,13 ± 4,4 -	96,39 ± 0,6	96,25 ± 0,9
Pendigits	96,39 ± 0,2	96,36 ± 0,3	97,21 ± 0,4 +	89,01 ± 0,5 -	**97,86 ± 0,3 +**	96,72 ± 0,5
CD-1	60,71 ± 3,0	58,26 ± 2,8	63,77 ± 3,7	61,40 ± 0,3	63,26 ± 4,1	**63,83 ± 1,2**
CD-2	65,55 ± 1,5	63,22 ± 1,6	**71,06 ± 5,1**	62,95 ± 5,1	70,26 ± 5,2	70,57 ± 2,4 +
CD-3	75,65 ± 2,4	71,64 ± 1,3 -	81,47 ± 1,7 +	69,24 ± 2,2 -	81,17 ± 1,9 +	**81,69 ± 0,9 +**
CD-4	**92,97 ± 0,9**	89,47 ± 0,8 -	92,58 ± 1,0	89,7 ± 2,9	92,80 ± 1,0	92,84 ± 1,1
CD-5	82,7 ± 2,8	78,71 ± 1,9	86,68 ± 1,9	72,31 ± 2,2 -	87,08 ± 1,8	**87,19 ± 2,1**
CD-6	57,78 ± 2,1	55,36 ± 2,3	**60,50 ± 2,3**	59,37 ± 3,7	60,28 ± 2,2	60,27 ± 1,2
CD-7	65,18 ± 1,0	60,68 ± 1,4 -	70,74 ± 1,8 +	65,90 ± 1,6	70,95 ± 1,9 +	**71,34 ± 1,2 +**
CD-8	75,57 ± 1,4	70,30 ± 1,7 -	**81,18 ± 2,1 +**	73,69 ± 1,9	81,11 ± 2,0+	80,98 ± 2,1 +
CD-9	93,00 ± 0,5	89,67 ± 1,4 -	93,89 ± 0,8	87,49 ± 2,9 -	93,87 ± 1,3	**93,98 ± 1,3 +**
CD-10	82,80 ± 1,7	78,45 ± 2,2 -	**86,25 ± 1,1 +**	73,89 ± 1,4 -	86,08 ± 1,5 +	85,83 ± 1,4
N. de Melhoras significativas		0	6	1	8	8
N. de pioras Significativas		7	3	11	1	2

Analisando os resultados da Tabela 4.3, para $S = 5$, da mesma forma que na Tabela 4.2, dentre os seis métodos os que obtiveram os piores resultados foram o C4.5 na sua versão sem poda, o qual não obteve o melhor resultado em nenhuma das 22 bases, e o AG-Sozinho, que obteve o melhor resultado em apenas uma base. O C4.5 com poda obteve os melhores resultados em duas das 22 bases. O C4.5 duplo obteve os melhores resultados em cinco bases. Dado que as taxas de acerto para as versões C4.5 com poda, C4.5 sem poda e o AG-Sozinho

independem do valor de S, não faz sentido repetir a descrição da análise anterior comparando os dois últimos com o C4.5 com poda. Analisando a *performance* do C4.5 duplo em relação ao C4.5 com poda, o primeiro foi significativamente melhor em seis bases de dados e significativamente pior em três dentre as 22 bases de dados. O C4.5/IB1 obteve taxas de acerto significativamente melhores que o C4.5 com poda em oito bases e significativamente piores em apenas uma base. O C4.5/AG-Grande-NS obteve as melhores taxas de acerto em 10 das 22 bases. Ao serem comparados os resultados deste método em relação ao C4.5 com poda, o C4.5/AG-Grande-NS obteve resultados significativamente melhores em oito bases e significativamente piores em apenas duas bases de dados.

Da mesma forma que na Figura 4.2, a Figura 4.3 mostra que, ao serem comparadas as taxas de acerto do C4.5/AG-Grande-NS com as taxas obtidas a partir do C4.5 com poda (S = 5), para a grande maioria das bases de dados houve uma melhora na taxa de acerto, chegando a pouco mais de 10% para as bases Hepatitis e Wave. Conforme já havia sido mencionado anteriormente, para o caso da base Segmentation houve uma piora significativa desta mesma taxa.

As Tabelas 4.4 e 4.5 apresentam os resultados obtidos nos experimentos com as 22 bases de dados para S = 10 e S = 15, respectivamente. A primeira coluna identifica as bases de dados, a segunda, a terceira e a quarta coluna mostram os resultados para as três versões do C4.5: com poda, sem poda e duplo, respectivamente. A quinta coluna apresenta os resultados para o algoritmo híbrido C4.5/IB1. A sexta coluna apresenta os resultados para o algoritmo híbrido C4.5/AG-Pequeno.E finalmente a sétima coluna apresenta os resultados obtidos para o algoritmo híbrido C4.5/AG-Grande-NS.

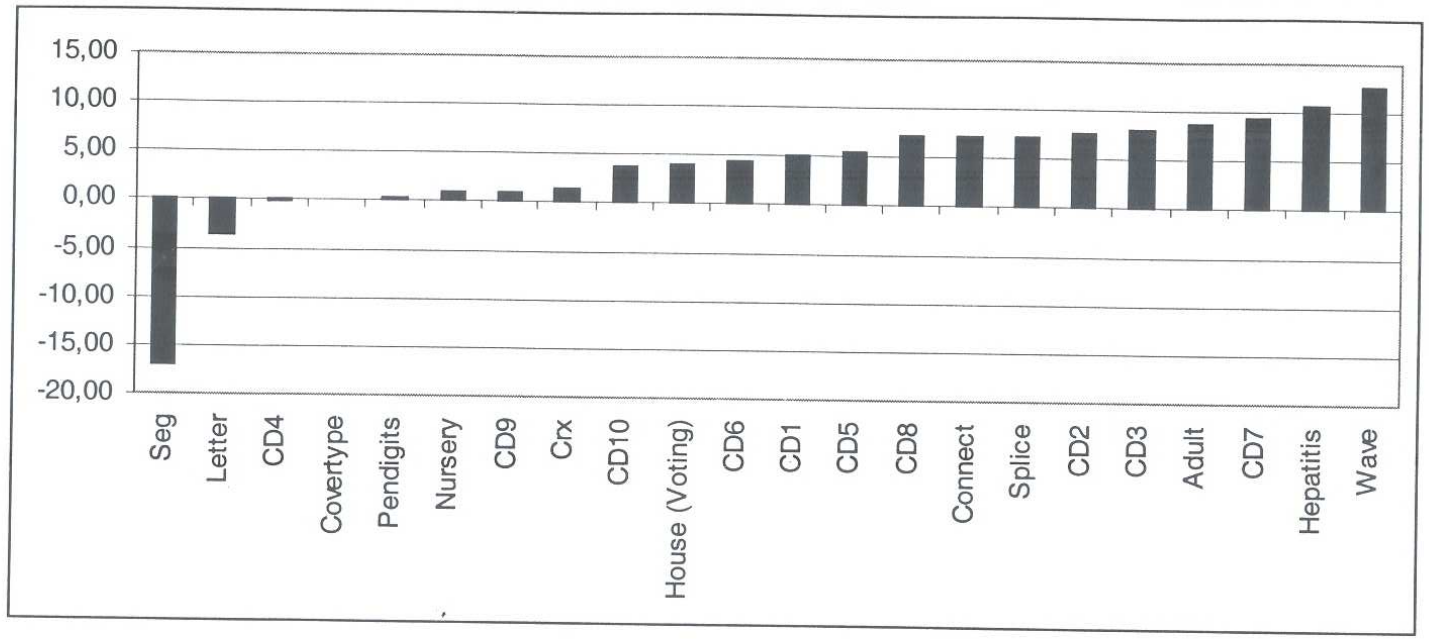


FIGURA 4.3 VARIAÇÃO (%) DA TAXA DE ACERTO DO C4.5/AG-GRANDE-NS EM RELAÇÃO À TAXA DE ACERTO DO C4.5 COM PODA PARA S = 5

TABELA 4.4 TAXA DE ACERTO (%) PARA $S = 10$

Base Dados	C4.5 com poda	C4.5 sem Poda	C4.5 duplo	AG-Sozinho	C4.5/IB1	C4.5/ AG-Pequeno	C4.5/AG-Grande-NS
Connect	72,60 ± 0,5	72,05 ± 0,5	76,19 ± 0,6 +	74,40 ± 0,6 +	**78,05 ± 0,5 +**	76,87 ± 0,0 +	76,95 ± 0,1 +
Adult	78,62 ± 0,5	77,00 ± 0,5 -	76,06 ± 0,5 –	78,90 ± 0,3	**80,94 ± 0,5 +**	80,62 ± 0,0 +	80,04 ± 0,1 +
Crx	91,79 ± 2,1	**92,45 ± 1,9**	90,78 ± 1,2	78,04 ± 1,2 -	90,61 ± 1,1	90,89 ± 1,3	91,66 ± 1,8
Hepatitis	80,78± 13,3	77,50 ± 11,3	82,36 ± 18,7	81,28 ± 11,2	88,91 ± 8,8	94,40 ± 6,2	**95,05 ± 7,2**
House	93,62 ± 3,2	93,50 ± 1,7	89,16 ± 8,0	**97,63 ± 1,6**	97,45 ± 1,7	96,80 ± 1,7	97,65 ± 2,0
Segmentat.	**96,86 ± 1,1**	96,30 ± 0,9	72,93 ± 5,5 –	72,42 ± 4,6 -	78,42 ± 1,2 -	79,00 ± 1,0 –	78,68 ± 1,1 –
Wave	75,78 ± 1,9	75,4 ± 2,1	64,93 ± 3,9 –	66,74 ± 4,6 -	83,24 ± 1,9 +	79,86 ± 4,2	**83,95 ± 3,0 +**
Splice	65,68 ± 1,3	66,54 ± 1,3	61,51 ± 6,6	60,26 ± 5,0	67,49 ± 6,5	67,04 ± 4,2	**70,70 ± 6,3**
Covertype	**71,61 ± 1,9**	70,34 ± 1,9	68,64 ± 14,8	65,40 ± 3,1 -	67,34 ± 16,8	69,43 ± 15,9	68,71 ± 1,3
Letter	86,40 ± 1,1	86,30 ± 1,1	82,77 ± 1,0 –	75,80 ± 0,3 -	**89,24 ± 1,1 +**	81,15 ± 0,0 –	79,24 ± 0,2 –
Nursery	95,40 ± 1,2	96,40 ± 0,9	**97,23 ± 1,0**	82,13 ± 4,4 -	97,13 ± 0,8	96,93 ± 0,6	96,77 ± 0,7
Pendigits	96,39 ± 0,2	96,36 ± 0,3	96,86 ± 0,4	89,01 ± 0,5 -	**97,91 ± 0,3 +**	94,96 ± 1,0 –	95,72 ± 0,9
CD-1	60,71 ± 3,0	58,26 ± 2,8	63,82 ± 5,2	61,40 ± 0,3	63,28 ± 4,3	**64,53 ± 4,5**	63,43 ± 1,4
CD-2	65,55 ± 1,5	63,22 ± 1,6	72,52 ± 5,9	62,95 ± 5,1	72,71 ± 5,7	73,52 ± 5,0 +	**73,77 ± 2,5 +**
CD-3	75,65 ± 2,4	71,64 ± 1,3 -	82,27 ± 1,3 +	69,24 ± 2,2 -	81,99 ± 2,2 +	83,16 ± 1,8 +	**84,15 ± 0,9 +**
CD-4	92,97 ± 0,9	89,47 ± 0,8 -	92,58 ± 1,0	89,7 ± 2,9	92,60 ± 0,9	**93,14 ± 0,9**	92,72 ± 1,0
CD-5	82,7 ± 2,8	78,71 ± 1,9	83,01 ± 1,9	72,31 ± 2,2 -	83,15 ± 1,8	**84,38 ± 2,1**	83,36 ± 2,1
CD-6	57,78 ± 2,1	55,36 ± 2,3	60,68 ± 3,2	59,37 ± 3,7	60,69 ± 2,9	60,91 ± 2,9	**61,69 ± 1,6 +**
CD-7	65,18 ± 1,0	60,68 ± 1,4 -	70,29 ± 2,4 +	65,90 ± 1,6	70,61 ± 2,4 +	**82,77 ± 2,0 +**	71,27 ± 1,6 +
CD-8	75,57 ± 1,4	70,30 ± 1,7 -	81,03 ± 1,9 +	73,69 ± 1,9	81,35 ± 1,6 +	81,78 ± 2,0 +	**82,63 ± 1,9 +**
CD-9	93,00 ± 0,5	89,67 ± 1,4 -	93,72 ± 1,2	87,49 ± 2,9 -	93,48 ± 1,3	87,33 ± 1,8 –	**93,80 ± 1,4**
CD-10	82,80 ± 1,7	78,45 ± 2,2 -	85,60 ± 1,4	73,89 ± 1,4 -	85,28 ± 1,3	86,76 ± 1,5 +	**86,88 ± 1,6 +**
N. de melhoras significat.		0	4	1	8	7	9
N. de pioras significat.		7	4	11	1	4	2

Analisando os resultados da Tabela 4.4, para $S = 10$, da mesma forma que nas Tabelas 4.2 e 4.3, dentre os sete métodos, os que obtiveram os piores resultados foram o C4.5 sem poda e o AG-Sozinho, ambos obtendo os melhores resultados em apenas uma das 22 bases de dados. O C4.5 com poda obteve os melhores resultados em duas das 22 bases. O C4.5 duplo obteve o melhor resultado em apenas uma base. Comparando a *performance* do C4.5 duplo em relação à do C4.5 com poda, o primeiro foi significativamente melhor em quatro e significativamente pior em quatro dentre as 22 bases de dados. O C4.5/IB1 foi significativamente melhor que o C4.5 com poda em oito bases e significativamente pior em apenas uma. Ao serem comparadas as taxas de acerto do C4.5/AG-Pequeno com as taxas do C4.5 com poda, o primeiro obteve resultados significativamente melhores em sete e significativamente piores em quatro bases. O C4.5/AG-Grande-NS obteve as melhores taxas de acerto em nove das 22 bases, sendo que, ao serem comparados os resultados deste método

em relação ao C4.5 com poda, o C4.5/AG-Grande-NS obteve resultados significativamente melhores em nove bases e significativamente piores em apenas duas bases de dados.

A Figura 4.4 mostra que, ao serem comparadas as taxas de acerto do C4.5/AG-Pequeno e do C4.5/AG-Grande-NS com as taxas de acerto do C4.5 com poda ($S = 10$), para a grande maioria das bases de dados houve um acréscimo na taxa de acerto, chegando a quase 30% para a base CD7 (C4.5/AG-Pequeno), quase 20% para a base Hepatitis (C4.5/AG-Pequeno e C4.5/AG-Grande-NS) e pouco mais de 10% para as bases Wave, CD2 e CD3 (C4.5/AG-Grande-NS). Para a base Segmentation houve uma piora significativa da mesma taxa de acerto, chegando a quase 20%, mas para nenhuma outra base a piora na taxa de acerto ultrapassou 10%.

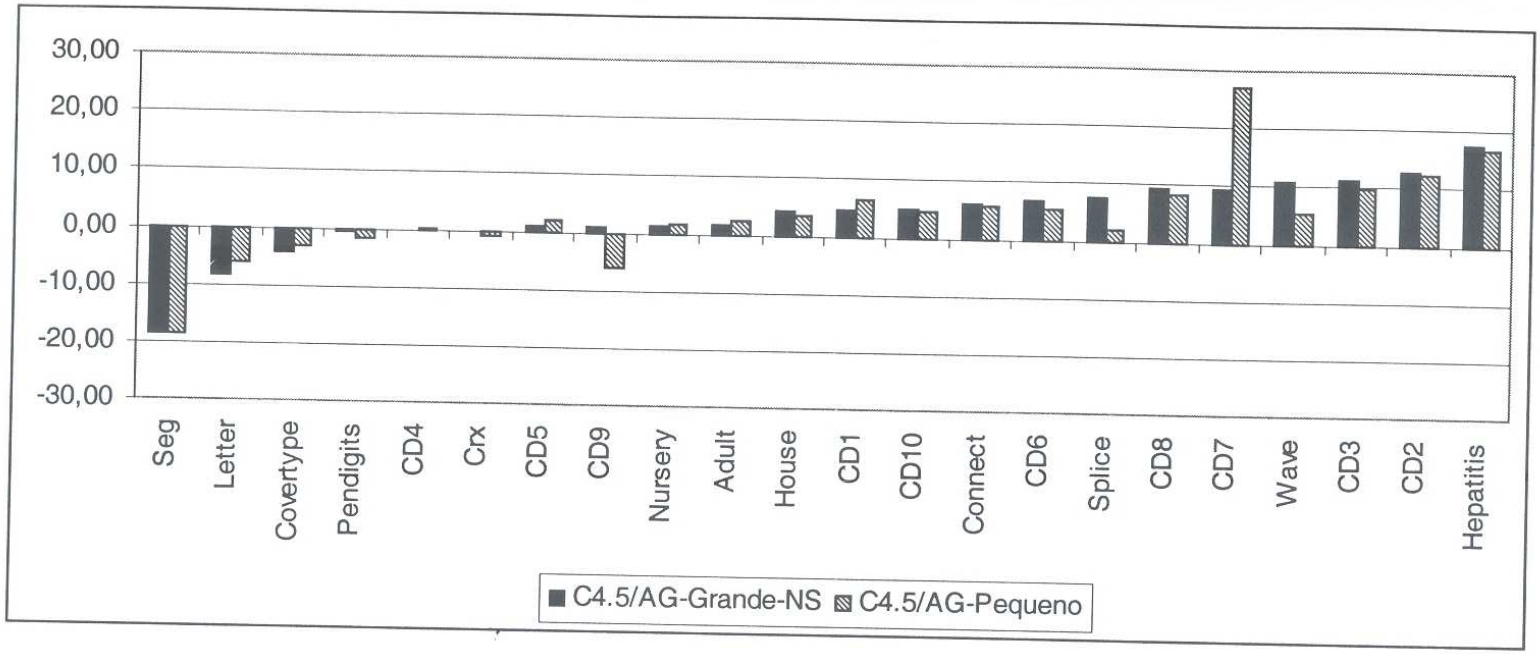


FIGURA 4.4 VARIAÇÃO (%) DA TAXA DE ACERTO DO C4.5/AG-PEQUENO E DO C4.5/AG-GRANDE-NS EM RELAÇÃO À TAXA DE ACERTO DO C4.5 COM PODA PARA S = 10

TABELA 4.5 TAXA DE ACERTO (%) PARA $S = 15$

Base Dados	C4.5 com poda	C4.5 sem poda	C4.5 duplo	AG-Sozinho	C4.5/IB1	C4.5/ AG-Pequeno	C4.5/AG-Grande-NS
Connect	72,60 ± 0,5	72,05 ± 0,5	74,95 ± 0,6 +	74,40 ± 0,6 +	**77,91 ± 0,5 +**	76,13 ± 0,0 +	76,01 ± 0,3 +
Adult	78,62 ± 0,5	77,00 ± 0,5 -	74,29 ± 0,5 -	78,90 ± 0,3	**80,37 ± 0,5 +**	79,97 ± 0,0 +	79,32 ± 0,2 +
Crx	91,79 ± 2,1	**92,45 ± 1,9**	90,02 ± 0,8	78,04 ± 1,2 -	89,46 ± 1,1	88,94 ± 2,3	90,40 ± 2,4
Hepatitis	80,78± 13,3	77,50 ± 11,3	66,16 ± 19,1	81,28 ± 11,2	**85,21 ± 7,8**	79,36 ± 23,4	82,52 ± 7,0
House	93,62 ± 3,2	93,50 ± 1,7	88,53 ± 8,4	**97,63 ± 1,6**	95,68 ± 1,7	94,88 ± 2,4	95,91 ± 2,3
Segment.	**96,86 ± 1,1**	96,30 ± 0,9	73,82 ± 5,8 -	72,42 ± 4,6 -	76,86 ± 1,7 -	77,00 ± 1,7 -	77,11 ± 1,9 -
Wave	75,78 ± 1,9	75,4 ± 2,1	65,53 ± 4,0 -	66,74 ± 4,6 -	82,43 ± 1,7 +	76,39 ± 5,0	**82,65 ± 3,7 +**
Splice	65,68 ± 1,3	66,54 ± 1,3	64,35 ± 4,7	60,26 ± 5,0	66,68 ± 5,5	66,53 ± 4,9	**70,62 ± 5,5**
Covertype	71,61 ± 1,9	70,34 ± 1,9	**68,87 ± 15,1**	65,40 ± 3,1 -	64,32 ± 17,1	68,51 ± 16,3	66,02 ± 1,3 –
Letter	86,40 ± 1,1	86,30 ± 1,1	81,35 ± 1,0 –	75,80 ± 0,3 -	**89,30 ± 1,1 +**	80,04 ± 0,0 –	76,38 ± 0,6 –
Nursery	95,40 ± 1,2	96,40 ± 0,9	**97,66 ± 0,8 +**	82,13 ± 4,4 -	97,26 ± 0,5 +	97,34 ± 1,2	96,64 ± 0,7
Pendigits	96,39 ± 0,2	96,36 ± 0,3	96,86 ± 0,4	89,01 ± 0,5 -	**97,98 ± 0,3 +**	95,71 ± 1,5	95,01 ± 1,2
CD-1	60,71 ± 3,0	58,26 ± 2,8	63,34 ± 4,9	61,40 ± 0,3	63,36 ± 4,0	63,68 ± 4,4	**63,92 ± 1,2**
CD-2	65,55 ± 1,5	63,22 ± 1,6	72,99 ± 4,8 +	62,95 ± 5,1	73,29 ± 4,8 +	74,36 ± 3,9 +	**74,75 ± 2,1 +**
CD-3	75,65 ± 2,4	71,64 ± 1,3 -	81,92 ± 2,7 +	69,24 ± 2,2 -	81,78 ± 2,3 +	83,00 ± 2,0 +	**83,06 ± 1,0 +**
CD-4	92,97 ± 0,9	89,47 ± 0,8 -	92,75 ± 1,4	89,7 ± 2,9	92,99 ± 1,3	93,28 ± 1,2	**93,48 ± 1,3**
CD-5	82,7 ± 2,8	78,71 ± 1,9	82,52 ± 2,0	72,31 ± 2,2 -	81,21 ± 1,8	82,61 ± 2,3	**82,81 ± 2,3**
CD-6	57,78 ± 2,1	55,36 ± 2,3	61,51 ± 3,1	59,37 ± 3,7	60,44 ± 2,5	61,78 ± 3,0	**62,07 ± 1,6 +**
CD-7	65,18 ± 1,0	60,68 ± 1,4 -	70,11 ± 2,6 +	65,90 ± 1,6	70,11 ± 2,6 +	**72,09 ± 3,1 +**	70,44 ± 2,0 +
CD-8	75,57 ± 1,4	70,30 ± 1,7 -	80,88 ± 1,3 +	73,69 ± 1,9	81,43 ± 1,2+	**83,20 ± 1,7 +**	81,79 ± 2,2 +
CD-9	93,00 ± 0,5	89,67 ± 1,4 -	93,60 ± 0,5	87,49 ± 2,9 -	93,58 ± 1,1	87,12 ± 1,6 –	**93,67 ± 1,3**
CD-10	82,80 ± 1,7	78,45 ± 2,2 -	85,59 ± 0,5 +	73,89 ± 1,4 -	84,92 ± 1,6	**86,71 ± 1,8 +**	85,70 ± 2,0
N. de melhora significat.		0	7	1	10	7	8
N. de piora significat.		7	4	11	1	3	3

Analisando os resultados da Tabela 4.5, para $S = 15$, da mesma forma que nas Tabelas 4.2, 4.3 e 4.4, dentre os sete métodos, os que obtiveram os piores resultados foram o C4.5 sem poda e o AG-Sozinho, obtendo os melhores resultados em apenas uma das 22 bases de dados. O C4.5 com poda obteve os melhores resultados em apenas uma das 22 bases. O C4.5 duplo obteve os melhores resultados em duas bases. Analisando a *performance* do C4.5 duplo em relação ao C4.5 com poda, o primeiro foi significativamente melhor em sete e significativamente pior em quatro dentre as 22 bases. O C4.5/IB1 foi significativamente melhor que o C4.5 com poda em 10 bases e significativamente pior em apenas uma dentre as 22 bases. O C4.5/AG-Pequeno obteve os melhores resultados em três das 22 bases. Ao serem comparadas as taxas de acerto do C4.5/AG-Pequeno com as taxas de acerto do C4.5 com poda, o primeiro obteve resultados significativamente melhores em sete e significativamente piores em três bases. O C4.5/AG-Grande-NS obteve as melhores taxas de acerto em nove das 22 bases. Ao serem comparados os resultados deste método em relação ao C4.5 com poda, o

C4.5/AG-Grande-NS obteve resultados significativamente melhores em oito bases e significativamente piores em apenas três bases.

A Figura 4.5 mostra que, ao serem comparadas as taxas de acerto do C4.5/AG-Pequeno e do C4.5/AG-Grande-NS com as taxas de acerto do C4.5 com poda ($S = 15$), para a maioria das bases de dados houve um acréscimo na taxa de acerto, sendo que nas bases CD2, CD3, CD7 e CD8 o acréscimo foi em torno de 10%. Para as bases Segmentation e Letter houve uma redução significativa das taxas de acerto, chegando a 20% para a primeira delas.

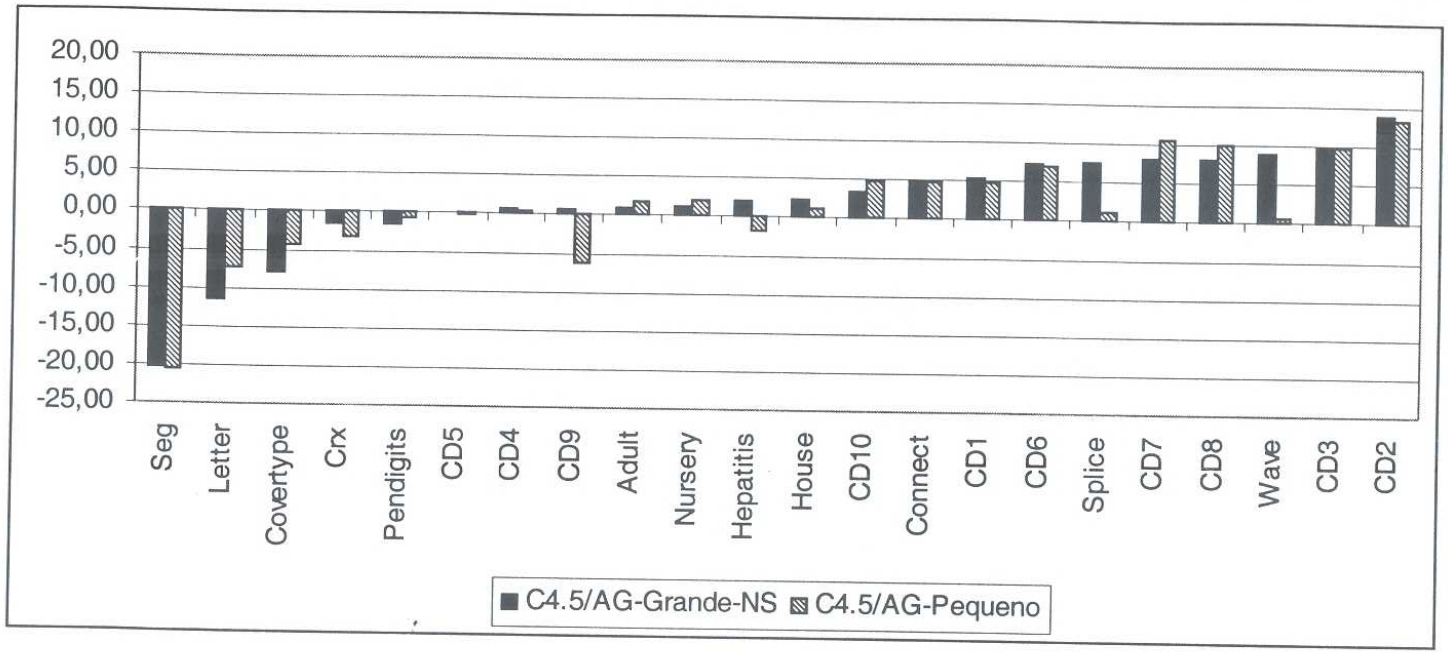


FIGURA 4.5 VARIAÇÃO (%) DA TAXA DE ACERTO DO C4.5/AG-PEQUENO E DO C4.5/AG-GRANDE-NS EM RELAÇÃO À TAXA DE ACERTO DO C4.5 COM PODA PARA S = 15

De forma geral, os resultados desta seção mostram que os sistemas híbridos C4.5/AG e C4.5/IB1, bem como o C4.5 duplo, obtiveram taxas de acerto bem melhores que o C4.5 com poda.

As Figuras 4.6, 4.7, 4.8 e 4.9 mostram a porcentagem de bases de dados onde cada método obteve a maior taxa de acerto, dentre os seis métodos comparados nas Tabelas 4.2 e 4.3 e os sete métodos comparados nas Tabelas 4.4 e 4.5, respectivamente.

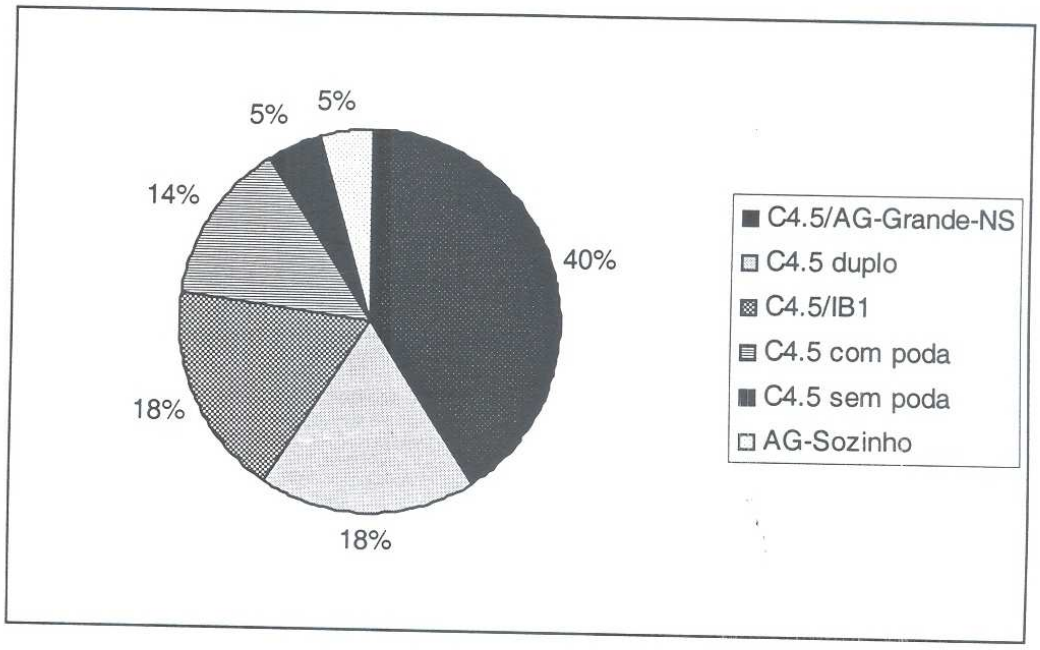


FIGURA 4.6 PORCENTAGEM DE BASES DE DADOS ONDE CADA MÉTODO OBTEVE A MELHOR TAXA DE ACERTO (S = 3)

Cabe ressaltar que o C4.5 sem poda não foi incluído no gráfico da Figura 4.7 devido ao fato que esse método não obteve a maior taxa de acerto para nenhuma das bases de dados. Além disso, o C4.5/AG-Pequeno não foi incluído nas Figuras 4.5 e 4.6 dado que esse algoritmo não foi aplicado para as definições de pequeno disjunto $S = 3$ e $S = 5$, conforme explicado anteriormente (seção 3.2.1).

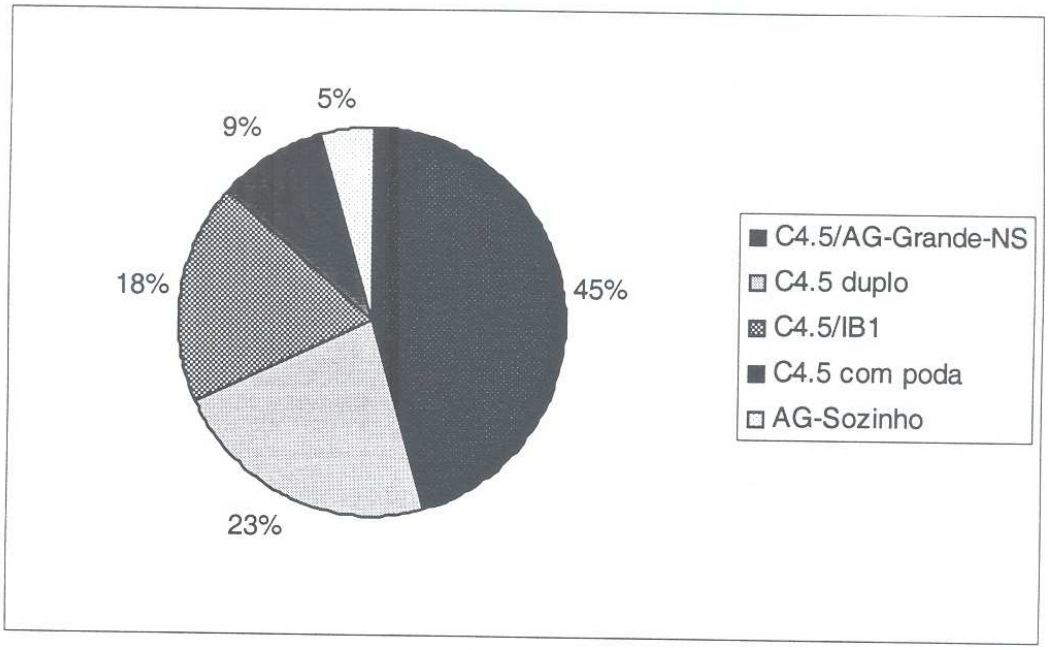


FIGURA 4.7 PORCENTAGEM DE BASES DE DADOS ONDE CADA MÉTODO OBTEVE A MELHOR TAXA DE ACERTO (S = 5)

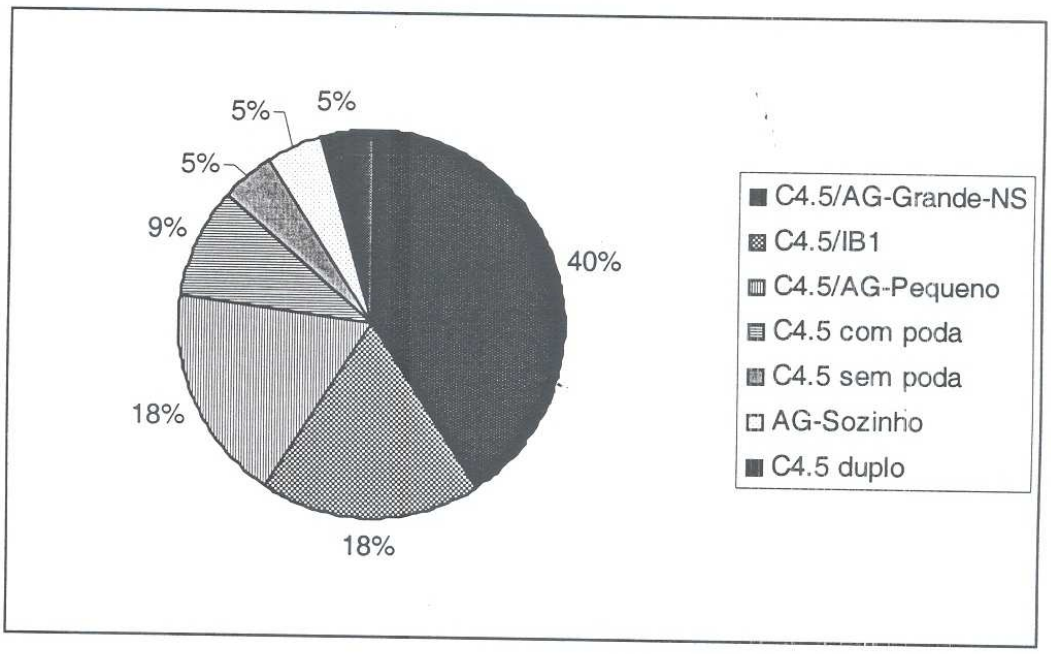


FIGURA 4.8 PORCENTAGEM DE BASES DE DADOS ONDE CADA MÉTODO OBTEVE A MELHOR TAXA DE ACERTO (S = 10)

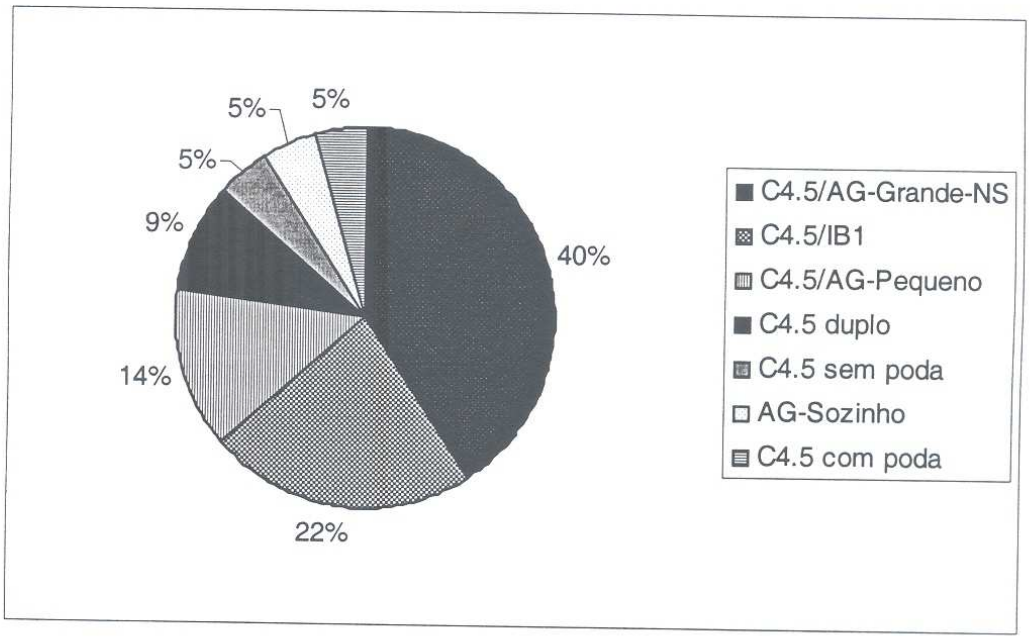


FIGURA 4.9 PORCENTAGEM DE BASES DE DADOS ONDE CADA MÉTODO OBTEVE A MELHOR TAXA DE ACERTO (S = 15)

É interessante notar que o método C4.5/AG-Grande-NS obteve o melhor resultado em torno de 40% das bases de dados, independentemente da definição de pequeno disjunto. O método C4.5/IB1 obteve o melhor resultado entre 18% e 23% das bases, dependendo da definição do pequeno disjunto. O método C4.5 duplo teve seu melhor desempenho para $S = 3$ e $S = 5$, onde ele obteve a melhor taxa de acerto em torno de 20% das bases de dados; já para $S = 10$ e $S = 15$ estes percentuais caíram para valores inferiores a 10%.

Portanto, comparando-se todos os algoritmos com relação ao percentual de bases de dados onde cada um obtém a maior taxa de acerto, pode-se afirmar que o melhor resultado foi claramente obtido pelo C.45/AG-Grande-NS. O segundo melhor resultado pode ser atribuído ao C4.5/IB1.

A Tabela 4.6 mostra em quantas bases de dados as taxas de acerto do C4.5/AG-Grande-NS foi significativamente melhor/pior que a dos outros métodos, para cada valor de *S*. Mais precisamente, em cada célula da tabela os resultados são apresentados na forma X / Y, onde X (Y) é o número de bases onde a taxa de acerto do C4.5/AG-Grande-NS foi significativamente melhor (pior) que a do método na correspondente coluna.

Analisando-se essa tabela (Tabela 4.6) pode-se concluir que em geral o C4.5/AG-Grande-NS constrói classificadores com taxas de acerto significativamente melhores que os algoritmos C4.5 com poda, C4.5 sem poda e o AG-Sozinho, para todos os quatro valores de *S*. Já comparando o algoritmo C4.5/AG-Grande-NS com os algoritmos C4.5/IB1 e C4.5/AG-Pequeno pode-se perceber que esta *performance* significativamente melhor do primeiro não se

repete, ou seja, este chega a ser significativamente pior que o C4.5/IB1 em três bases de dados, tanto para $S = 10$ quanto para $S = 15$. Vale lembrar que nesta comparação só está sendo considerada uma das características desejáveis do conhecimento descoberto, a taxa de acerto.

Em geral, os algoritmos competitivos em relação ao C4.5/AG-Grande-NS, considerando-se apenas o critério de maximização da taxa de acerto, são o C4.5 duplo, o C4.5/IB1, o C4.5/AG-Pequeno.

TABELA 4.6 RESULTADOS SIGNIFICATIVAMENTE MELHORES/PIORES DO ALGORITMO C4.5/AG-GRANDE-NS EM RELAÇÃO AOS DEMAIS ALGORITMOS

Valores de S	C4.5 com poda	C4.5 sem poda	C4.5 duplo	AG-Sozinho	C4.5/IB1	C4.5/AG-Pequeno
3	9 / 2	14 / 2	4 / 1	15 / 0	0 / 1	-
5	8 / 2	13 / 2	3 / 0	17 / 0	0 / 1	-
10	9 / 2	14 / 2	3 / 1	17 / 0	0 / 3	1 / 2
15	8 / 3	13 / 3	3 / 1	13 / 0	0 / 3	1 / 2

Naturalmente, ao se considerar também o critério de maximização de simplicidade das regras descobertas, o AG-Grande-NS passa a ser bem mais vantajoso que o C4.5 duplo, o C4.5/IB1 e o C4.5/AG-Pequeno, conforme resultados a serem mostrado e discutidos na seção 4.6 e o fato de que o componente IB1 do método C4.5/IB1 não descobre regras generalizando os dados.

4.6 SIMPLICIDADE

Foi avaliada a simplicidade do conjunto de regras descobertas para cada um dos seguintes seis métodos: C4.5 com poda, C4.5 sem poda, C4.5 duplo, C4.5-AG-Pequeno, C4.5/AG-Grande-NS e AG-Sozinho. Não foi considerado este quesito para o método C4.5/IB1 dado o fato do conhecimento extraído não ser representado na forma de regras do tipo SE-ENTÃO. Para cada um dos demais métodos, foi medido o número de regras e o tamanho (número de condições) médio das regras. A forma como estas medidas foram obtidas é descrita a seguir.

No caso do C4.5 com poda e C4.5 sem poda, a árvore induzida é inicialmente convertida em um conjunto de regras. O número de regras é o número de nós folha na árvore. Cada caminho a partir do nó raiz até o nó folha torna-se uma regra, cujo antecedente é constituído pelos testes sobre os atributos previsores que se encontram naquele caminho e cujo conseqüente é determinado pela classe predita pelo nó folha. A partir deste momento foi aplicado um pós-processamento simples sobre cada regra, com o objetivo de reduzir o

tamanho sem alterar a semântica da regra. Todas as condições de um mesmo atributo foram convertidas em uma única condição equivalente. Por exemplo, supondo que o C4.5 tenha construído uma regra com um caminho que contemple dois testes com o atributo idade: idade > 21, idade > 25. Estes dois testes são convertidos em uma única condição: idade > 25. Após a execução deste pós-processamento, foi calculado o número médio de condições por regra através da divisão do número total de condições de todas as regras pelo número total de regras.

Para o C4.5 duplo, C4.5/AG-Pequeno e C4.5/AG-Grande-NS, o cálculo do número de regras é realizado da seguinte forma: o primeiro passo é identificar na árvore gerada pelo C4.5 os nós folha que correspondem a grande disjuntos. Cada um destes percursos de grande disjunto é convertido em uma regra, conforme descrição anterior. Obtém-se então r_{grande}, onde r_{grande} representa o número de regras de grandes disjuntos.

O segundo passo é identificar o número de regras ($r_{pequeno}$) e o número de condições descobertas para os pequenos disjuntos. No caso do C4.5 duplo, estes valores são obtidos após a conversão da árvore induzida através da segunda execução do C4.5 (sobre o segundo conjunto de treinamento) em um conjunto de regras, usando o mesmo método adotado no caso das outras duas versões do C4.5 (descrito anteriormente). No caso do C4.5/AG-Pequeno e C4.5/AG-Grande-NS, o número de regras de pequenos disjuntos ($r_{pequeno}$) e o número de condições são dados pelos números de regras e condições descobertas pela versão do AG componente do sistema.

Para cada um desses três métodos o número total de regras descobertas é simplesmente o número de regras de grandes disjuntos (r_{grande}) – que é o mesmo para os três métodos – mais o número de regras de pequenos disjuntos ($r_{pequeno}$) descobertas pelo método específico. O número médio de condições é obtido pela divisão do número total de condições (tanto para as regras de grande e de pequeno disjunto) pelo número total de regras (r_{grande} + $r_{pequeno}$).

No caso do AG-Sozinho o procedimento se limita a identificar o número de regras descobertas, bem como o número médio de condições, dado que para este classificador não existe um tratamento distinto para regras de pequeno e de grande disjunto como nos algoritmos C4.5/AG-Pequeno, C4.5/AG-Grande-NS e C4.5 duplo.

Conforme explicado anteriormente, o sistema C4.5/AG-Pequeno descobre um número maior de regras que o C4.5 com poda, tendo em vista que, para cada pequeno disjunto identificado pela árvore gerada pelo C4.5 com poda, o componente AG do C4.5/AG-Pequeno

descobre c regras, onde c é o número de classes. Entretanto, o C4.5 duplo e o C4.5/AG-Grande-NS não possuem esta desvantagem. A princípio, estes dois métodos podem descobrir um número total de regras ($r_{grande} + r_{pequeno}$) consideravelmente menor que o número de regras descobertas pelo C4.5 com poda. Isto se deve ao fato destes métodos usarem um segundo conjunto de treinamento relativamente grande, oferecendo uma oportunidade para que se descubra poucas regras (cada uma com uma ampla cobertura) para cobrir os exemplos de pequenos disjuntos, se comparado com o C4.5 com poda.

Os resultados em relação a simplicidade (número e tamanho médio das regras descobertas), para $S = 3$, podem ser observados na Tabela 4.7. Os mesmos resultados correspondentes a $S = 5$, $S = 10$ e $S = 15$ constam das Tabelas 4.8, 4.9 e 4.10, respectivamente. Estes resultados foram obtidos a partir dos mesmos experimentos usados para a obtenção da taxa de acerto, mostrados nas Tabelas 4.2, 4.3, 4.4 e 4.5. Da mesma forma que nas tabelas da seção anterior, nas Tabelas 4.7, 4.8, 4.9 e 4.10, o melhor resultado entre os seis algoritmos é marcado em negrito (para cada um dos dois quesitos de simplicidade), e o número após o símbolo "±" é o desvio padrão. Nas colunas de resultados está indicado, para cada base de dados, se o valor obtido para o número de regras e o número médio de condições das regras descobertas pelo método referenciado na coluna é significativamente diferente do valor obtido através do C4.5 com poda. Relembrando, os casos onde um dos algoritmos obteve um resultado significativamente melhor (pior) que o C4.5 com poda é indicado pelo símbolo "+" ("-").

O símbolo ">", o qual ocorre nos resultados do C4.5 sem poda nas bases de dados Connect e Adult, indica que o valor real do resultado é maior que o valor expresso na célula, mas não pode ser obtido exatamente. Esta situação decorre do fato de que nessas bases a árvore não-podada tem mais de 99 subárvores, e o algoritmo C4.5 não consegue gerar um arquivo de saída (em formato texto) representando a árvore de decisão resultante com um número superior a 99 subárvores.

TABELA 4.7 SIMPLICIDADE (NÚMERO E TAMANHO MÉDIO DAS REGRAS DESCOBERTAS) PARA $S = 3$

Base Dados	C4.5 com poda		C4.5 sem poda		C4.5 duplo		AG-Sozinho		C4.5/AG-Grande-NS	
	#regras	tamanho	#regras	tamanho	#regras	Tamanho	#regras	tamanho	#regras	tamanho
Connect	479 ± 3,90	5,53 ± 0,04	>3864±31,48 -	5,54 ± 0,04	354 ± 2,8 +	5,40± 0,04+	**22,0 ± 3,59 +**	**3,18 ± 0,23 +**	237 ± 1,73 +	5,35±0,02 +
Adult	1115 ± 7,43	6,76 ± 0,04	>2411±16,06 -	6,82± 0,04 -	898 ± 5,98 +	6,99± 0,04+	**16,1 ± 2,67 +**	**2,12 ± 0,13 +**	575 ± 2,31 +	6,63 ± 0,01
Crx	55,1 ± 4,1	5,18 ± 0,34	72,60 ± 4,33 -	5,49 ± 0,42	36,4 ± 3,6 +	4,66 ± 0,35	**7,36 ± 1,47 +**	**2,90 ± 0,29 +**	33,79 ± 2,84 +	4,91 ± 0,36
Hepatitis	8,8 ± 2,5	**1,30 ± 0,13**	13,30 ± 1,89 -	1,43 ± 0,20	6,1 ± 1,7	1,32 ± 0,06	5,84 ± 1,09	2,82 ± 0,29 -	**5,30 ± 1,68**	1,43 ± 0,23
House	10,1 ± 3,6	2,97 ± 0,28	17,80 ± 6,46	3,52 ± 0,24	9,4 ± 2,5	**1,21** ± 0,22	**3,48 ± 1,15 +**	1,85 ± 0,59	7,81 ± 2,39	2,74 ± 0,18
Segment	45,0 ± 4,8	2,33 ± 0,15	52,30 ± 3,95	2,49 ± 0,11	38,8 ± 3,4	**2,29** ± 0,21	**21,96± 2,11 +**	3,03 ± 0,18 -	37,07 ± 2,72 +	2,41 ± 0,12
Wave	354,1 ± 8,5	5,30 ± 0,31	397,9 ± 10,35 -	5,40 ± 0,29	261,3 ± 6,0 +	5,08 ± 0,31	**14,65 ± 2,37 +**	**1,65 ± 0,23 +**	213,38 ± 5,63+	5,09 ± 0,28
Splice	120 ± 8,5	2,60 ± 0,10	178,4 ± 5,78 -	2,81 ± 0,09 -	51,3 ± 6,0 +	**2,44 ± 0,13**	**14,60± 4,13 +**	2,77 ± 0,38	46,96 ± 3,85 +	2,47 ± 0,10
Covertype	787,6 ± 25,50	6,93 ± 0,09	1025,7 ± 30,1 -	7,23± 0,08 -	477,8± 25,7 +	6,82 ±0,11	**24,07 ± 4,93 +**	**3,10 ± 0,38 +**	464,72± 14,63 +	6,82 ± 0,10
Letter	808,0 ± 10,43	3,95 ± 0,04	927,0±11,96 -	4,24± 0,04 -	710 ± 8,65 +	3,97 ± 0,04	**136,5 ± 5,44 +**	**3,19 ± 0,11 +**	595,90 ± 4,25 +	3,81 ± 0,02
Nursery	318,6 ± 26,76	5,94 ± 0,05	609,3 ± 24,33 -	6,34± 0,05 -	231,4 ± 33,1 +	5,11±0,33 +	**25,04 ± 5,34 +**	**2,15 ± 0,21 +**	227,80± 32,60 +	5,13 ± 0,29
Pendigits	181,0 ± 5,44	4,45 ± 0,03	211,6 ± 7,11 -	4,63± 0,08 -	150,7 ± 4,1 +	4,04±0,13 +	**41,71 ± 4,07 +**	**3,36 ± 0,15 +**	142,80 ± 2,97+	4,06±0,12 +
CD-1	624,3 ± 27,64	4,96 ± 0,14	1383,6± 38,5 -	5,18 ± 0,14	353,4 ± 25,8 +	4,83 ± 0,13	**16,9 ± 2,77 +**	**3,06 ± 0,25 +**	312,61± 12,74 +	4,91 ± 0,13
CD-2	598,9 ± 20,89	5,95 ± 0,13	1175,9± 28,1 -	6,12 ± 0,09	304,4 ± 38,0 +	5,75 ± 0,11	**17,31± 3,45 +**	**2,97 ± 0,24 +**	274,30± 14,79 +	5,83 ± 0,11
CD-3	386,1 ± 23,86	4,41 ± 0,09	922,5± 24,23 -	4,74± 0,09 -	222,1± 20,15 +	4,30 ± 0,12	**17,25 ± 2,47 +**	**2,89 ± 0,23 +**	186,33 ± 9,59 +	4,31 ± 0,08
CD-4	46,1 ± 3,79	3,51 ± 0,36	411,0± 25,32 -	4,98± 0,21 -	30,40 ± 4,7 +	3,24 ± 0,24	**10,18 ± 2,80 +**	**2,98 ± 0,33 +**	26,90 ± 3,46 +	3,40 ± 0,27
CD-5	221,0 ± 7,37	4,56 ± 0,19	737,6± 22,46 -	4,77± 0,19	148,1 ± 20,6 +	4,34 ± 0,19	**11,73 ± 2,82 +**	**2,83 + 0,35 +**	122,45± 15,27 +	4,40 ± 0,18
CD-6	665,6 ± 55,35	5,10 ± 0,27	1424,7±25,36 -	5,56± 0,14 -	386,1 ± 22,4 +	4,99 ± 0,24	**17,4 ± 2,72 +**	**2,97 ± 0,25 +**	340,12± 30,19 +	5,06 ± 0,25
CD-7	577,2 ± 37,01	5,88 ± 0,11	1237,1±29,06 -	6,11± 0,10 -	313,3 ± 25,1 +	5,59 ± 0,13	**17,69 ± 1,83 +**	**3,00 ± 0,33 +**	268,04± 14,38 +	5,78 ± 0,09
CD-8	346,2 ± 33,50	4,19 ± 0,19	966,7± 27,13 -	4,79± 0,15 -	204,9 ± 20,6 +	4,14 ± 0,16	**16,81 ± 2,57 +**	**2,88 ± 0,26 +**	168,31± 15,87 +	4,12 ± 0,18
CD-9	41,50 ± 7,60	3,48 ± 0,62	388,0± 19,26 -	4,98± 0,28 -	28,8 ± 4,5 +	3,15 ± 0,56	**9,39 ± 2,77 +**	**2,94 ± 0,38**	25,04 ± 3,96 +	3,34 ± 0,46
CD-10	209,2 ± 31,33	4,44 ± 0,43	753,3±27,89 -	4,85 ± 0,25	138,2 ± 16,0 +	4,23 ± 0,37	**11,96 ± 2,67 +**	**2,93 ± 0,33 +**	111,37± 12,35 +	4,32 ± 0,41
N. de Melhoras Significat.			0	0	19	4	21	17	20	2
N. de pioras Significat.			20	12	0	0	0	2	0	0

As Tabelas 4.7, 4.8, 4.9 e 4.10 apresentam na primeira coluna a identificação das bases de dados. A partir da segunda coluna as colunas mostram os resultados para cada um dos algoritmos comparados de duas em duas colunas, uma contendo o número médio de regras (#regras) e a outra o tamanho (número de condições) médio das regras descobertas. Desta forma a segunda e terceira colunas apresentam os resultados para o C4.5 com poda, a quarta e quinta colunas apresentam os resultados para o C4.5 sem poda, e assim sucessivamente para os algoritmos C4.5 duplo, AG-Sozinho, C4.5/AG-Pequeno e C4.5/AG-Grande-NS. Lembrando que o C4.5/AG-Pequeno só foi testado para os valores de $S = 10$ e $S = 15$ (Tabelas 4.9 e 4.10). Finalmente, as duas últimas linhas das Tabelas 4.7, 4.8, 4.9 e 4.10 sumarizam os resultados, mostrando o número de melhoras e pioras significativas para cada algoritmo em comparação com o *baseline* C4.5 com poda.

Analisando a Tabela 4.7, pode ser observado que os algoritmos C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS descobrem classificadores com um número consideravelmente menor de regras do que o C4.5 com poda para todas as 22 bases de dados. Em 60 dos 66 casos (22 bases de dados * 3 algoritmos) a diferença entre o número de regras descobertas pelo C4.5 duplo, AG-Sozinho e o C4.5/AG-Grande-NS e o número de regras descobertas pelo C4.5 com poda é significativa, ou seja, não ocorre sobreposição dos intervalos dos respectivos desvios padrões. As exceções são: a base Hepatitis nos três algoritmos, a base House-votes para o C4.5 duplo e C4.5/AG-Grande-NS, e a base Segmentation para o C4.5 duplo. Nestes casos de exceção, o número de regras descobertas pelo C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS, embora menor, não foi significativamente diferente do número de regras descobertas pelo C4.5 com poda. A redução do C4.5 duplo em relação ao C4.5 com poda é superior a 40% em sete das 22 bases de dados, a redução do AG-Sozinho em relação ao C4.5 com poda é superior a 90% em 13 das 22 bases de dados e a redução do C.45/AG-Grande-NS redução é superior a 40% em 13 das 22 bases de dados.

Como era de se esperar, o número de regras descobertas pelo C4.5 sem poda é bem superior ao número de regras descobertas pelas outras duas versões do C4.5 (com poda e duplo), para todos os 44 casos. Na comparação entre o número de regras descobertas pelo C4.5 com poda e pelo C4.5 duplo, o C4.5 duplo obteve classificadores com um menor número de regras para todas as 22 bases de dados. Valendo destacar que a redução chega a 57% para a base Splice e 49% para a base CD-2 .

Na grande maioria das bases de dados o AG-Sozinho descobriu um número de regras muito menor que os demais métodos. Porém, deve-se lembrar que a taxa de acerto do algoritmo AG-Sozinho não demonstrou ser competitiva para bases com existência de pequenos disjuntos (Tabelas 4.2, 4.3, 4.4 e 4.5).

É possível observar que houve uma redução no número de regras geradas pelo AG-Grande-NS em relação ao C4.5 duplo para todas as 22 bases de dados. A maior redução (35%) ocorreu na base Adult e a menor redução (1,5%) na base Nursery.

Ainda analisando a Tabela 4.7 quanto à questão do tamanho médio das regras obtidas pelos algoritmos comparados, o AG-Sozinho obteve o menor tamanho médio em 18 das 22 bases de dados. Em relação ao C4.5 com poda, o C4.5 duplo obteve quatro e o AG-Grande-NS duas melhoras significativas, sem nenhuma piora significativa.

Comparando especificamente o C4.5 duplo e o C4.5 com poda, é possível perceber que para apenas três bases não houve uma redução no tamanho médio das regras descobertas, a saber as bases Letter, Hepatitis e Adult, onde foram registradas melhoras de 0,5%, 1,5% e 3,4%, respectivamente. Entre as bases onde houveram pioras, a maior ocorreu na base House-votes (59%).

Comparando o C4.5/AG-Grande-NS e o C4.5 com poda, em apenas duas bases não houve redução no tamanho médio das regras descobertas, a saber as bases Segmentation e Hepatitis, onde foram registradas melhoras de 3,4% e 10%, respectivamente. Entre as bases onde houveram pioras, a maior ocorreu na base Nursery (13,6%).

Comparando o C4.5/AG-Grande-NS e o C4.5 duplo, em apenas quatro bases houve redução no tamanho médio das regras descobertas, sendo a maior redução registrada na base Adult (5,15%).

As Figuras 4.10, 4.12, 4.14 e 4.16 mostram a porcentagem de aumento ou diminuição do número médio de regras descobertas pelos algoritmos C4.5 sem poda, C4.5 duplo, C4.5/AG-Pequeno, C4.5/AG-Grande-NS e AG-Sozinho em relação ao número de regras descobertas pelo C4.5 com poda para $S = 3$, $S = 5$, $S = 10$ e $S = 15$, respectivamente.

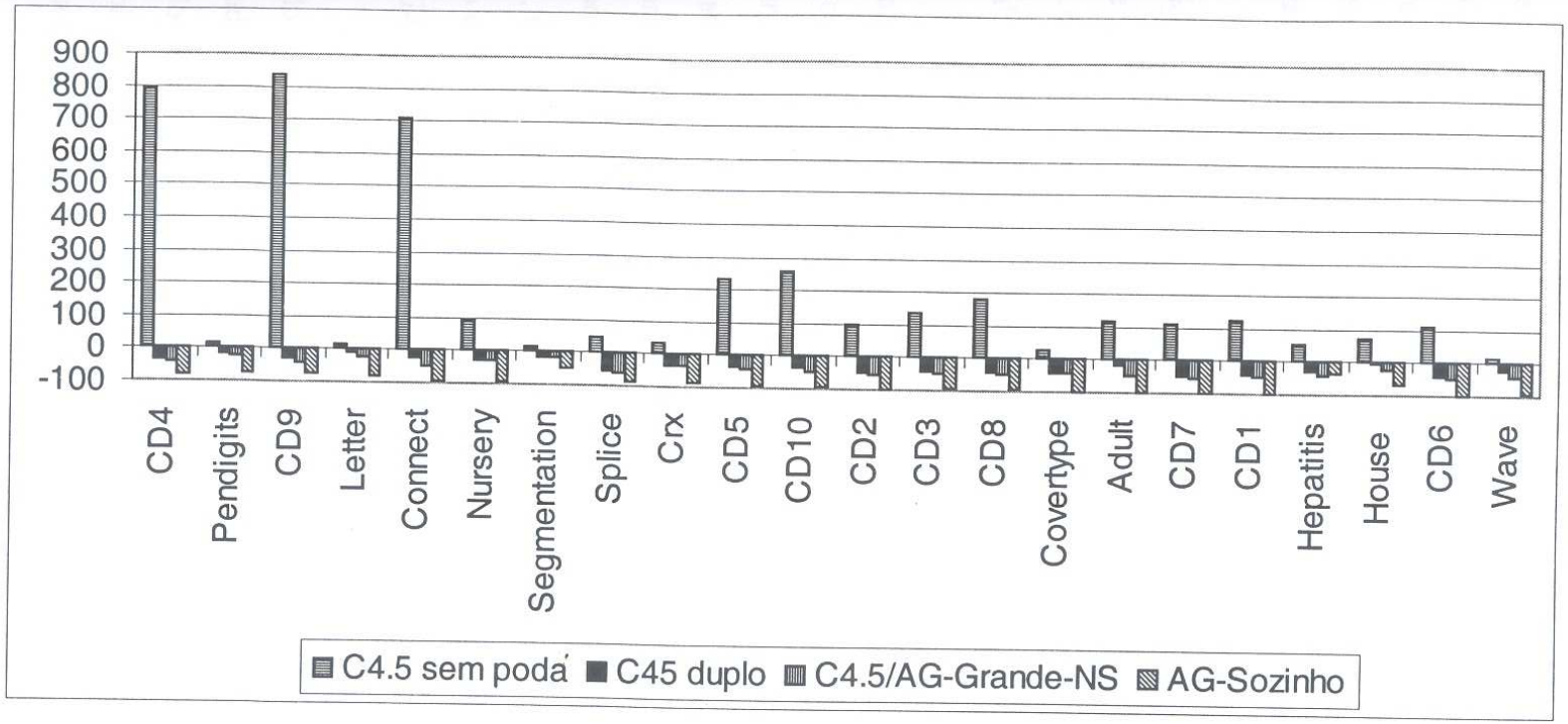


FIGURA 4.10 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA ($S = 3$)

A partir da Figura 4.10 pode-se confirmar que para todas as bases de dados o conjunto de regras descobertas pelo C4.5 sem poda é maior que o conjunto de regras descoberto pelo C4.5 com poda, esse aumento chega a estar entre 700% a 800% para as bases CD4, CD9 e Connect. Já em relação ao conjunto de regras descoberto pelos demais algoritmos, todos descobrem um conjunto de regras menor que o conjunto descoberto pelo C4.5 com poda. Esta redução na cardinalidade do conjunto chega a ser quase 100%, comparando o AG-Sozinho com o C4.5 com poda, para várias bases de dados.

As Figuras 4.11, 4.13, 4.15 e 4.17 mostram a porcentagem de aumento ou diminuição do tamanho médio das regras em relação ao tamanho médio das regras descobertas pelos algoritmos C4.5 sem poda, C4.5 duplo, C4.5/AG-Pequeno, C4.5/AG-Grande-NS e AG-Sozinho em relação ao tamanho médio das regras descobertas pelo C4.5 com poda para $S = 3$, $S = 5$, $S = 10$ e $S = 15$, respectivamente.

A partir da Figura 4.11 pode-se confirmar que não só quanto ao número de regras, mas também no quesito tamanho médio das regras descobertas, o C4.5 sem poda descobre regras com tamanho médio maior que o tamanho médio das regras descobertas pelo C4.5 com poda para todas as bases. Esse aumento varia de $\cong$ 0.1% (Connect) a $\cong$ 40% (CD4). Já em relação ao tamanho médio das regras descobertas pelos demais algoritmos em comparação ao C4.5 com poda, a grande maioria descobre regras com um tamanho médio menor que o algoritmo C4.5 com poda. Esta redução chega a ser $\cong$ 50% para o AG-Sozinho (Connect, Nursery, CRX, CD2, Covertype, Adult, CD7, CD6 e Wave). Para as bases onde ocorre um aumento do tamanho médio das regras descobertas, essa ocorre entre $\cong$ 3% (Segmentation – C4.5/AG-Grande-NS) e $\cong$ 115% (Hepatitis – AG-Sozinho).

Da mesma forma que para $S = 3$ (Tabela 4.7), para $S = 5$ (Tabela 4.8) pode ser observado que os algoritmos C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS descobrem classificadores com um número consideravelmente menor de regras para todas as 22 bases de dados. Novamente em 60 dos 66 casos a diferença entre o número de regras descobertas pelo C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS e o número de regras descobertas pelo C4.5 com poda é significativa.

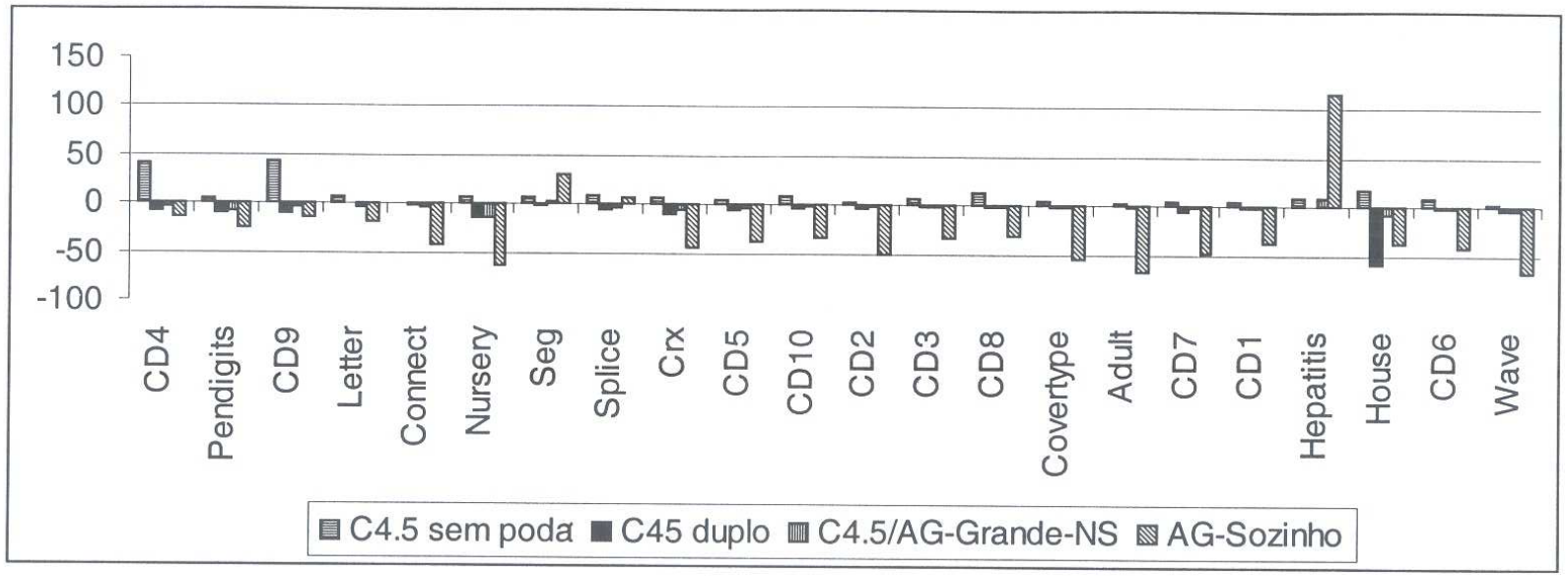


FIGURA 4.11 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA ($S = 3$)

TABELA 4.8 SIMPLICIDADE (NÚMERO E TAMANHO MÉDIO DAS REGRAS DESCOBERTAS) PARA $S = 5$

Base de Dados	C4.5 com Poda		C4.5 sem Poda		C4.5 duplo		AG-Sozinho		C4.5/AG-Grande-NS	
	#regras	tamanho	#regras	tamanho	#regras	Tamanho	#regras	tamanho	#regras	tamanho
Connect	479 ± 3,90	5,53 ± 0,04	>3864±31,48 -	5,54 ± 0,04	360± 2,92 +	5,03 ± 0,04 +	**22,0 ± 3,59 +**	**3,18 ± 0,23 +**	155,2± 1,23 +	5,26 ± 0,04
Adult	1115 ± 7,43	6,76 ± 0,04	>2411±16,06 -	6,82± 0,04 -	1010± 7,72 +	7,28 ± 0,05 -	**16,1 ± 2,67 +**	**2,12 ± 0,13 +**	389,2± 1,55 +	6,38 ± 0,01 +
Crx	55,1 ± 4,1	5,18 ± 0,34	72,60± 4,33 -	5,49 ± 0,42	32,3±4,72 +	4,42 ± 0,28	**7,36 ± 1,47 +**	**2,90 ± 0,29 +**	26,54±2,97 +	4,65 ± 0,30
Hepatitis	8,8 ± 2,5	**1,30 ± 0,13**	13,30 ± 1,89 -	1,43 ± 0,20	5,7 ± 0,95	1,54 ± 0,06 -	5,84 ± 1,09	2,82 ± 0,29 -	**4,98±0,77 +**	1,65 ± 0,43
House	10,1 ± 3,6	2,97 ± 0,28	17,80 ± 6,46	3,52 ± 0,24	9,2 ± 2,97	**1,21**± 0,21 +	**3,48 ± 1,15 +**	1,85 ± 0,59 +	7,36 ± 1,94	2,62 ± 0,20
Segment,	45,0 ± 4,8	2,33 ± 0,15	52,30 ± 3,95	2,49 ± 0,11	37,0 ± 3,65	**2,35** ± 0,19	**21,96±2,11 +**	3,03 ± 0,18 -	32,21±2,07 +	2,45 ± 0,11
Wave	354,1 ± 8,5	5,30 ± 0,31	397,9 ± 10,35 -	5,40 ± 0,29	234,6±7,0 +	5,10 ± 0,28	**14,65± 2,37 +**	**1,65 ± 0,23 +**	147,42±8,97+	4,96 ± 0,25
Splice	120 ± 8,5	2,60 ± 0,10	178,4 ± 5,78 -	2,81 ± 0,09 -	49,0±5,5 +	**2,48 ± 0,29**	**14,60± 4,13 +**	2,77 ± 0,38	38,97±2,58 +	2,36 ± 0,12 +
Covertype	787,6 ± 25,50	6,93 ± 0,09	1025,7 ±30,1 -	7,23± 0,08 -	349,3±27,7+	6,72 ± 0,12	**24,07± 4,93 +**	**3,10 ± 0,38 +**	319,07±2,7 +	6,68 ± 0,10 +
Letter	808,0 ± 10,43	3,95 ± 0,04	927,0±11,96 -	4,24± 0,04 -	671,0± 8,65 +	3,97 ± 0,04	**136,5± 5,44 +**	**3,19 ± 0,11 +**	465,4±5,08 +	3,81 ± 0,02 +
Nursery	318,6 ± 26,76	5,94 ± 0,05	609,3 ± 24,33 -	6,34± 0,05 -	201,4±26,29+	4,92 ± 0,37	**25,04± 5,34 +**	**2,15 ± 0,21 +**	195,3±24,5 +	5,13 ± 0,29 +
Pendigits	181,0 ± 5,44	4,45 ± 0,03	211,6 ± 7,11 -	4,63± 0,08 -	138,8±4,1 +	4,05 ± 0,21 +	**41,71± 4,07 +**	**3,36 ± 0,15 +**	122,8±2,81 +	4,06 ± 0,12 +
CD-1	624,3 ± 27,64	4,96 ± 0,14	1383,6± 38,5 -	5,18 ± 0,14	274,1±32,63+	4,74 ± 0,18	**16,9 ± 2,77 +**	**3,06 ± 0,25 +**	193,69±9,7 +	4,91 ± 0,13
CD-2	598,9 ± 20,89	5,95 ± 0,13	1175,9± 28,1 -	6,12 ± 0,09	225,4±48,0 +	5,53± 0,20 +	**17,31± 3,45 +**	**2,97 ± 0,24 +**	165,26±9,5 +	5,83 ± 0,11
CD-3	386,1 ± 23,86	4,41 ± 0,09	922,5± 24,23 -	4,74± 0,09 -	177,3±27,9 +	4,21 ± 0,12	**17,25± 2,47 +**	**2,89 ± 0,23 +**	114,6±8,61 +	4,31 ± 0,08
CD-4	46,1 ± 3,79	3,51 ± 0,36	411,0± 25,32 -	4,98± 0,21 -	26,0±4,95 +	3,16 ± 0,35	**10,18± 2,80 +**	**2,98 ± 0,33**	20,04±3,04 +	3,40 ± 0,27
CD-5	221,0 ± 7,37	4,56 ± 0,19	737,6± 22,46 -	4,77 ± 0,19	126,3±9,72 +	4,34 ± 0,23	**11,73± 2,82 +**	**2,83 + 0,35 +**	79,70±4,91 +	4,40 ± 0,18
CD-6	665,6 ± 55,35	5,10 ± 0,27	1424,7±25,36 -	5,56± 0,14 -	296,4±31,55+	4,75 ± 0,31	**17,4 ± 2,72 +**	**2,97 ± 0,25 +**	214,88±23,2+	6,06 ± 0,25 -
CD-7	577,2 ± 37,01	5,88 ± 0,11	1237,1±29,06 -	6,11± 0,10 -	252,6±27,7 +	5,27 ± 0,20 +	**17,69± 1,83 +**	**3,00 ± 0,33 +**	159,69±6,07+	5,78 ± 0,09
CD-8	346,2 ± 33,50	4,19 ± 0,19	966,7± 27,13 -	4,79± 0,15 -	170,1±15,49+	4,14 ± 0,30	**16,81± 2,57 +**	**2,88 ± 0,26 +**	102,97±0,78+	4,12 ± 0,18
CD-9	41,50 ± 7,60	3,48 ± 0,62	388,0± 19,26 -	4,98± 0,28 -	22,9±5,32 +	**2,92 ± 0,59**	**9,39 ± 2,77 +**	2,94 ± 0,38	17,28±2,72 +	3,34 ± 0,46
CD-10	209,2 ± 31,33	4,44 ± 0,43	753,3±27,89 -	4,85 ± 0,25	125,4±15,83+	4,11 ± 0,34	**11,96± 2,67 +**	**2,93 ± 0,33 +**	71,16±9,26 +	4,32 ± 0,41
N. de Melhoras significat.			0	0	19	5	21	17	20	6
N. de Pioras significat.			20	12	0	2	0	2	0	1

A redução do C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS em relação ao C4.5 com poda é superior a 40% em 13 das 22 bases de dados, é superior a 90% em 13 das 22 bases de dados, e superior a 60% em 11 das 22 bases de dados, respectivamente.

Ainda sobre a Tabela 4.8, o AG-Sozinho obteve o menor tamanho médio em 18 das 22 bases de dados. Em relação ao C4.5 com poda o C4.5 duplo obteve cinco e o AG-Grande-NS obteve seis melhoras significativas, obtendo apenas duas e uma pioras significativas, respectivamente.

Comparando especificamente o C4.5 duplo e o C4.5 com poda é possível perceber que para apenas quatro bases não houve uma redução no tamanho médio da regras descobertas, ou seja houve aumento, a saber as bases Letter (0,5%), Segmentation (8,5%), Adult (7,7%) e Hepatitis (18%). Entre as bases onde houveram reduções, a maior redução ocorreu para a base House-votes (59%).

Comparando o C4.5/AG-Grande-NS e o C4.5 com poda, em apenas três bases não houve redução no tamanho médio das regras descobertas, sendo a maior redução observada para a base Hepatitis (26,9%) e, dentre as bases onde houve reduções, a maior delas (13,6%) ocorreu na base Nursery (13,6%).

Comparando o C4.5/AG-Grande-NS com o C4.5 duplo, em seis bases houve redução no tamanho médio das regras descobertas, sendo a maior redução registrada para a base Adult (12,4%).

Analisando a Figura 4.12 pode-se concluir que também para $S = 5$, as mesmas observações que foram feitas para $S = 3$ (Figura 4.10) se repetem, quanto a cardinalidade o conjunto de regras descoberta pelos diversos algoritmos. O C4.5 duplo obteve redução no número de regras descobertas para todas as bases de dados variando entre $\cong$ 9% e $\cong$ 60%. O C4.5/AG-Grande-NS obteve redução no número de regras descobertas para todas as bases de dados variando entre $\cong$ 30% e $\cong$ 70%. O AG-Sozinho obteve redução no número de regras descobertas para todas as bases de dados variando entre $\cong$ 30% e $\cong$ 98%.

Da mesma forma que no quesito cardinalidade das regras descobertas não houveram alterações significativas comparando os resultados para os valores de $S = 3$ e $S = 5$, para o item tamanho médio das regras descobertas, a Figura 4.13 mostra que também não são percebidas alterações significativas em relação a Figura 4.11.

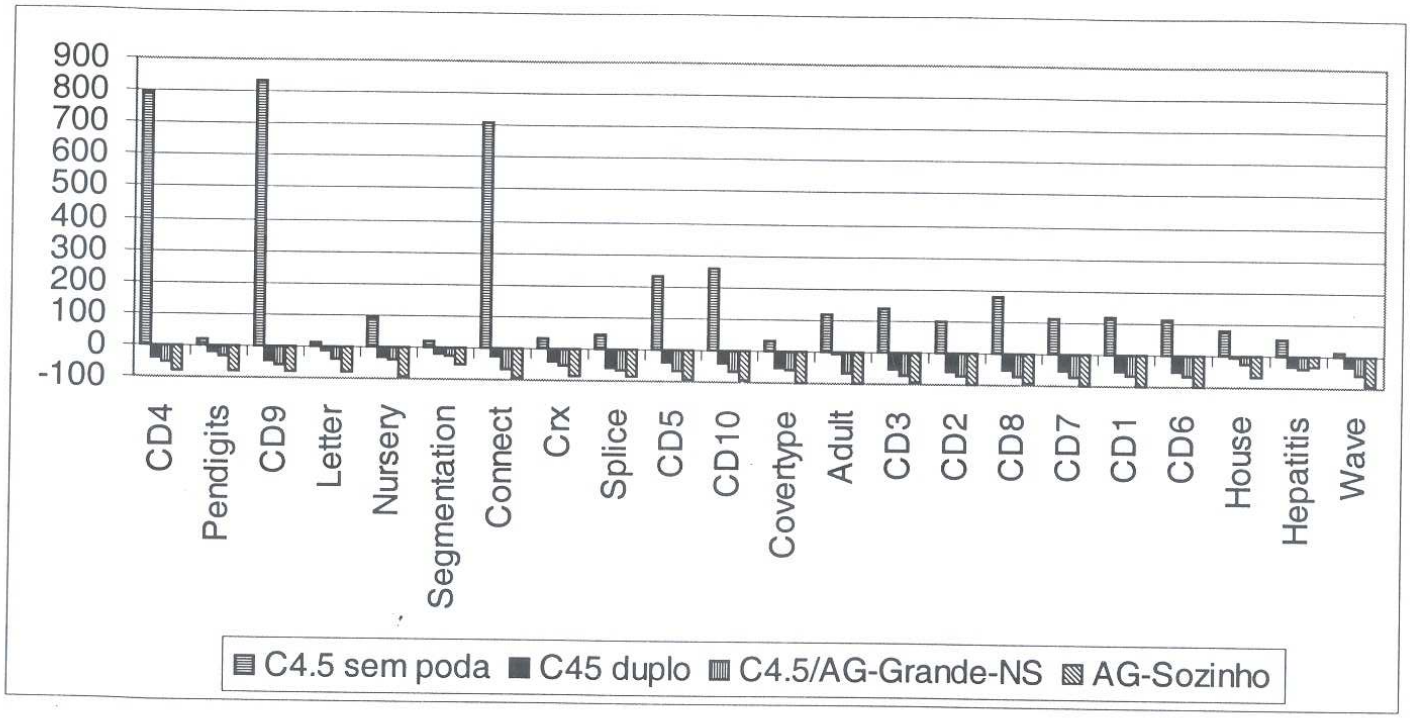


FIGURA 4.12 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA ($S = 5$)

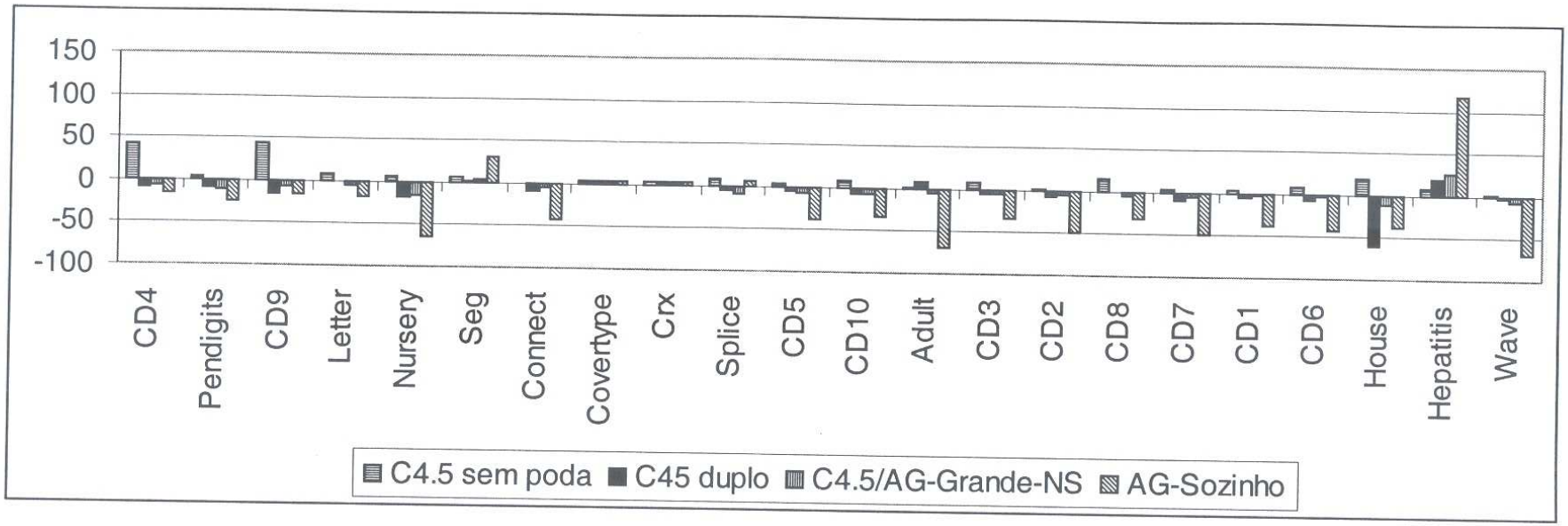


FIGURA 4.13 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA ($S = 5$)

TABELA 4.9 SIMPLICIDADE (NÚMERO E TAMANHO MÉDIO DAS REGRAS DESCOBERTAS) PARA $S = 10$

Base de Dados	C4.5 com poda		C4.5 sem poda		C4.5 duplo		AG-Sozinho		C4.5/AG – Pequeno		C4.5/AG-Grande-NS	
	#regras	tamanho	#regras	tamanho	#regras	tamanho	#regras	Tamanho	#regras	tamanho	#regras	tamanho
Connect	479 ± 3,90	5,53 ± 0,0	>3864±31,48 -	5,54 ± 0,04	505±4,11 -	5,35 ± 0,04 +	**22,0 ± 3,59 +**	**3,18 ± 0,23 +**	990,0 ± 0,0 -	6,61±0,06 -	92,0 ± 3,4 +	4,98 ± 0,08 +
Adult	1115± 7,43	6,76 ± 0,0	>2411±16,06 -	6,82±0,04 -	1245±8,29 -	7,53 ± 0,05 +	**16,1 ± 2,67 +**	**2,12 ± 0,13 +**	1814 ± 0,0 -	5,43±0,01 +	192,9 ± 0,7 +	5,90 ± 0,01 +
Crx	55,1 ± 4,1	5,18 ± 0,3	72,60 ± 4,33 -	5,49 ± 0,42	31,0±3,5 +	4,36 ± 0,42 +	**7,36 ± 1,47 +**	**2,90 ± 0,29 +**	83,1 ± 1,31 -	5,07 ± 0,18	19,6 ± 1,7 +	4,08 ± 0,34 +
Hepatitis	8,8 ± 2,5	**1,30 ± 0,1**	13,30 ± 1,89 -	1,43 ± 0,20	5,3 ± 1,3	2,58 ± 0,72 -	5,84 ± 1,09	2,82 ± 0,29 -	9,7 ± 1,8 -	3,07±0,83 -	**3,9 ± 0,9 +**	2,14 ± 0,35 -
House	10,1 ± 3,6	2,97 ± 0,3	17,80 ± 6,46	3,52 ± 0,24	7,8 ± 3,2	**1,48 ± 0,35 +**	**3,48 ± 1,15 +**	1,85 ± 0,59	15,8 ± 3,9 -	3,53 ± 0,52	6,6 ± 1,5	2,16 ± 0,29 +
Segment.	45,0 ± 4,8	2,33 ± 0,2	52,30 ± 3,95	2,49 ± 0,11	35,1±4,1 +	**2,23 ± 0,14**	**21,96± 2,11 +**	3,03 ± 0,18 -	71,20 ± 9,62 -	4,31±0,34 -	28,7 ± 2,3 +	2,48 ± 0,15
Wave	354,1± 8,5	5,30 ± 0,3	397,90±10,35 -	5,40 ± 0,29	234,7±7,4 +	5,43 ± 0,45	**14,65 ± 2,37 +**	**1,65 ± 0,23 +**	632,9 ± 12,6 -	6,48±0,30 -	89,8 ± 6,6 +	4,69 ± 0,26 +
Splice	120 ± 8,5	2,60 ± 0,1	178,40±5,78 -	2,81± 0,09 -	51,8±7,8 +	2,57 ± 0,20	**14,60± 4,13 +**	2,77 ± 0,38	228,48± 18,7 -	5,37±0,51 -	32,1 ± 2,3 +	**2,19 ± 0,11 +**
Covertype	787,6± 25,5	6,93 ± 0,1	1025,7± 30,1 -	7,23±0,08 -	238,5±34,9 +	6,65 ± 0,31	**24,07 ± 4,93 +**	**3,10 ± 0,38 +**	1421,3±43,21 -	6,93± 0,62	168,5±12,2 +	6,24 ± 0,16 +
Letter	808,0± 10,4	3,95 ± 0,0	927,0±11,96 -	4,24±0,04 -	701,0±9,04 +	4,00 ± 0,05	**136,5 ± 5,44 +**	**3,19 ± 0,11 +**	1499,0± 0,0 -	3,54±0,04 +	317,8±10,2 +	3,67 ± 0,04 +
Nursery	318,6± 26,7	5,94 ± 0,1	609,3 ± 24,33 -	6,34±0,05 -	140,9±11,3 +	5,07 ± 0,35 +	**25,04 ± 5,34 +**	**2,15 ± 0,21 +**	427,7 ± 69,22 -	4,52±0,34 +	126,78±8,3 +	5,08 ± 0,35 +
Pendigits	181,0± 5,44	4,45 ± 0,0	211,6 ± 7,11 -	4,63±0,08 -	120,4±5,15 +	4,12 ± 0,21 +	**41,71 ± 4,07 +**	**3,36 ± 0,15 +**	322,1 ± 26,19 -	5,02±0,15 -	96,02±3,0 +	3,98 ± 0,15 +
CD-1	624,3± 27,6	4,96 ± 0,1	1383,6± 38,5 -	5,18 ± 0,14	250,5±40,4 +	4,78 ± 0,47	**16,9 ± 2,77 +**	**3,06 ± 0,25 +**	756,6 ± 36,56 -	6,34±0,32 -	92,24±8,0 +	4,56 ± 0,14 +
CD-2	598,9± 20,8	5,95 ± 0,1	1175,9± 28,1 -	6,12 ± 0,09	193,8±52,3 +	5,33 ± 0,34 +	**17,31± 3,45 +**	**2,97 ± 0,24 +**	602,5 ± 34,17	6,49±0,29 -	71,63±5,7 +	5,28 ± 0,17 +
CD-3	386,1± 23,8	4,41 ± 0,1	922,5± 24,23 -	4,74±0,09 -	163,1±46,8 +	4,47 ± 0,34	**17,25 ± 2,47 +**	**2,89 ± 0,23 +**	466,7 ±50,02 -	5,75±0,22 -	54,07±6,3 +	3,93 ± 0,13 +
CD-4	46,1 ± 3,79	3,51 ± 0,4	411,0± 25,32 -	4,98±0,21 -	26,7±5,62 +	3,40 ± 0,49	**10,18 ± 2,80 +**	**2,98 ± 0,33 +**	58,4 ± 11,29	4,33±0,33 -	12,91±1,9 +	3,08 ± 0,26
CD-5	221,0± 7,37	4,56 ± 0,2	737,6± 22,46 -	4,77 ± 0,19	118,9±12,4 +	4,54 ± 0,34	**11,73 ± 2,82 +**	**2,83 + 0,35 +**	410,0± 19,68 -	5,37±0,25 -	39,03±4,6 +	3,96 ± 0,20 +
CD-6	665,6± 55,3	5,10 ± 0,3	1424,7±25,36 -	5,56±0,14 -	268,1±38,6 +	4,57 ± 0,28	**17,4 ± 2,72 +**	**2,97 ± 0,25 +**	890,3± 68,14 -	6,38±0,40 -	105,3±11,4 +	4,68 ± 0,25
CD-7	577,2± 37,0	5,88 ± 0,1	1237,1±29,06 -	6,11±0,10 -	243,4±30,3 +	5,36 ± 0,40 +	**17,69 ± 1,83 +**	**3,00 ± 0,33 +**	727,0± 35,81 -	6,55±0,30 -	74,03±6,7 +	5,27 ± 0,12 +
CD-8	346,2± 33,5	4,19 ± 0,2	966,7± 27,13 -	4,79±0,15 -	165,8±33,2 +	4,31 ± 0,47	**16,81 ± 2,57 +**	**2,88 ± 0,26 +**	435,5±53,82 -	5,54±0,36 -	46,63±6,5 +	3,71 ± 0,18 +
CD-9	41,50± 7,60	3,48 ± 0,6	388,0± 19,26 -	4,98±0,28 -	25,70±7,2 +	3,27 ± 0,58	**9,39 ± 2,77 +**	**2,94 ± 0,38**	64,20± 9,96 -	4,52± 0,42	12,53± 1,9 +	3,04 ± 0,31
CD-10	209,2± 31,3	4,44 ± 0,4	753,3±27,89 -	4,85 ± 0,25	119,2±13,5 +	4,28 ± 0,49	**11,96 ± 2,67 +**	**2,93 ± 0,33 +**	362,8±20,02 -	6,78±0,67 -	32,86±4,0 +	3,84 ± 0,39
N. de Melhoras Significat.			0	0	18	8	21	17	0	3	21	16
N. de Pioras Significat.			20	12	2	1	0	2	20	15	0	1

A Tabela 4.9 mostra os dados sobre a simplicidade para $S = 10$. Da mesma forma que para $S = 3$ (Tabela 4.7) e $S = 5$ (Tabela 4.8), os algoritmos C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS descobrem classificadores com um número consideravelmente menor de regras para todas as 22 bases de dados. Em 60 dos 66 casos a diferença entre o número de regras descobertas pelos três algoritmos e o número de regras descobertas pelo C4.5 com poda é significativa.

Com respeito ao número de regras descobertas pelo sistema C4.5/AG-Pequeno em relação aos demais métodos, o resultado é o pior para as 22 bases de dados. A razão para isso foi explicada na seção 3.2.1. Na comparação C4.5/AG-Pequeno com o C4.5 com poda, o número de regras do primeiro chega a ser o dobro do segundo. Já na comparação do C4.5/AG-Grande-NS em relação ao C4.5 com poda a redução chega a ser mais de 80% em 10 bases.

Sobre os resultados relativos a tamanho médio das regras descobertas, a Tabela 4.9 mostra que o AG-Sozinho, da mesma forma que para $S = 3$ e $S = 5$ (Tabelas 4.7 e 4.8), obteve o menor tamanho médio em 18 das 22 bases de dados. Em relação ao C4.5 com poda, o C4.5 duplo obteve oito e o AG-Grande-NS 16 melhoras significativas, sendo que para ambos ocorreu uma piora significativa na base Hepatitis.

Comparando especificamente o C4.5 duplo e o C4.5 com poda, é possível perceber que em seis bases não houve uma redução no tamanho médio das regras descobertas, a saber as bases Letter, CD-3, Wave, CD-8, Adult e Hepatitis, onde foram registrados aumentos de 1,2%, 1,3%, 2,4%, 2,8%, 11,4% e 98,5%, respectivamente. Entre as bases onde houveram reduções, a maior redução ocorreu na base House-votes (50%).

Comparando o C4.5/AG-Grande-NS com o C4.5 com poda, em apenas duas bases houve redução no tamanho médio da regras descobertas, a saber as bases Segmentation e Hepatitis, onde foram registrados aumentos de 6,43% e 64,6%, respectivamente. Entre as bases onde houveram reduções no tamanho médio das regras descobertas, a maior redução ocorreu na base House-votes (27,3%).

Comparando o C4.5/AG-Grande-NS e o C4.5 duplo, em apenas quatro bases não houve redução no tamanho médio das regras descobertas, e o maior aumento foi registrado para a base House-votes (45,4%). Entre as bases que obtiveram reduções, pode-se destacar a base Adult, onde houve uma redução de 21,6% no tamanho médio das regras descobertas.

Analisando a Figura 4.14 pode-se concluir que também para $S = 10$, as mesmas observações que foram feitas para $S = 3$ (Figura 4.10) e $S = 5$ (Figura 4.12), quanto a

cardinalidade o conjunto de regras descoberta pelos diversos algoritmos, se repetem, com exceção do algoritmo C4.5 para a base Connect onde ocorreu um pequeno aumento da cardinalidade do conjunto de regras descobertas em relação ao C4.5 com poda, ao invés de uma redução. Vale ressaltar que para $S = 10$ são relatados resultados para o algoritmo C4.5/AG-Pequeno, o qual apresentou aumento da cardinalidade em relação ao C4.5 com poda para todas as bases de dados testadas. Esse aumento varia entre ≅1% (CD2) até ≅100% (Connect).

A partir da Figura 4.15 pode-se verificar que não só quanto ao número de regras, mas também no quesito tamanho médio das regras descobertas, o C4.5 sem poda descobre regras com tamanho médio maior que o tamanho médio das regras descobertas pelo C4.5 com poda para todas as bases. Esse aumento varia de ≅ 0.1% a ≅ 40%. Em relação ao tamanho médio das regras descobertas pelos demais algoritmos em relação ao C4.5 com poda, pode-se concluir que:

- o C4.5 duplo reduz em 19 das 22 bases, sendo que essas reduções variam entre ≅ 1% e ≅60%. Os aumentos variam entre ≅ 1% e ≅ 80%;
- o C4.5/AG-Pequeno reduz em apenas 3 das 22 bases (entre ≅ 10% e ≅ 27%) e os aumentos variam entre ≅ 4% e ≅ 100%;
- o C4.5/Ag-Grande-NS reduz em 19 das 22 bases (entre ≅ 10% e ≅ 60%) e os aumentos variam entre ≅ 4% e ≅12%; e
- o AG-Sozinho reduz em 19 das 22 bases (entre ≅ 3% e ≅ 65%) e os aumentos variam entre ≅ 20% e ≅30%.

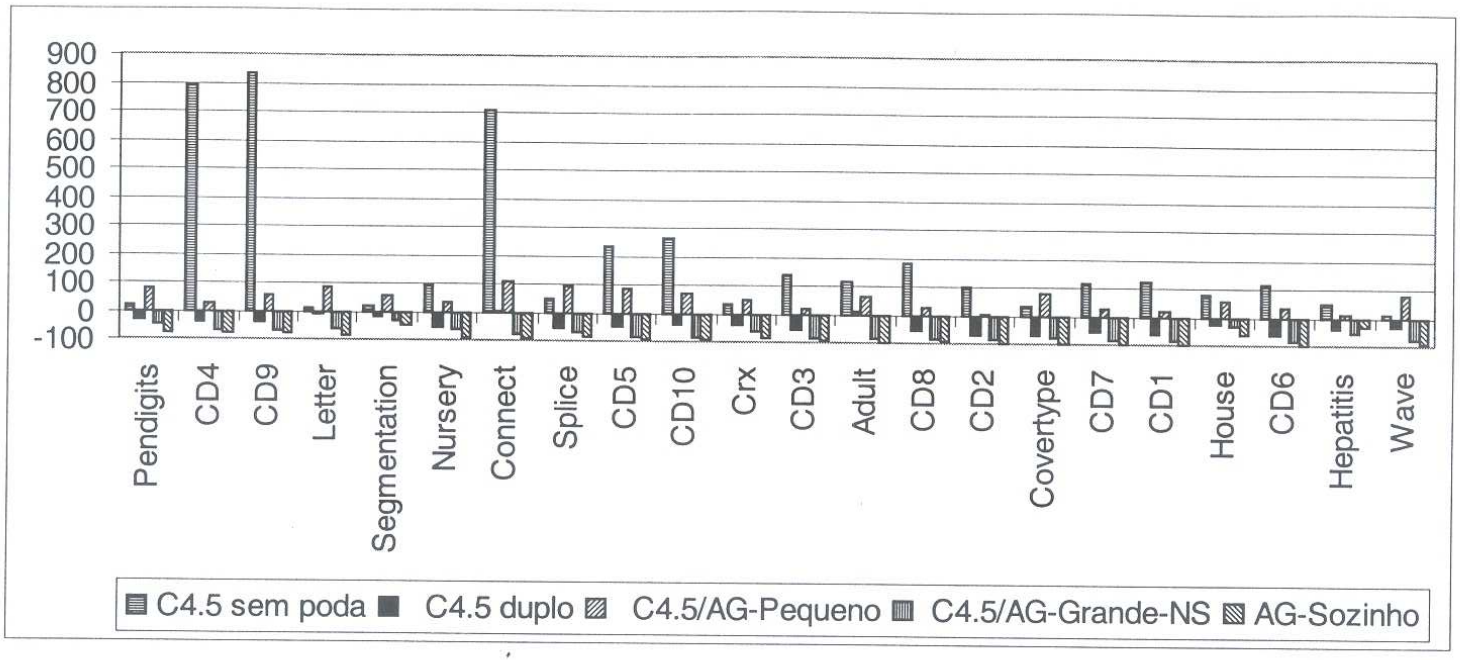


FIGURA 4.14 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA ($S = 10$)

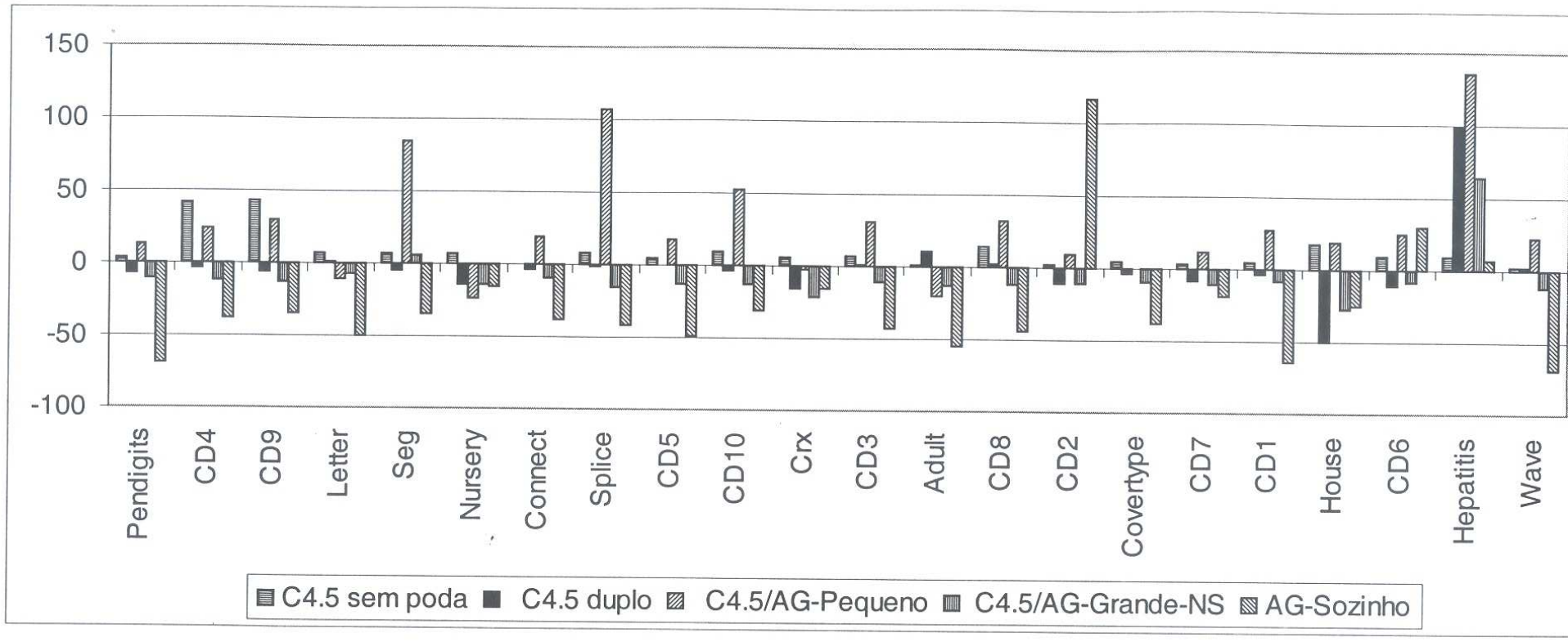


FIGURA 4.15 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA ($S = 10$)

TABELA 4.10 SIMPLICIDADE (NÚMERO E TAMANHO MÉDIO DAS REGRAS DESCOBERTAS) PARA $S = 15$

Base de Dados	C4.5 com poda		C4.5 sem poda		C4.5 duplo		AG-Sozinho		C4.5/ AG-Pequeno		C4.5/AG-Grande-NS	
	#regras	Tamanho	#regras	tamanho	#regras	tamanho	#regras	tamanho	#regras	tamanho	#regras	tamanho
Connect	479 ± 0,5	5,53 ± 0,4	>3864±0,6 -	5,54± 0,6	645 ± 5,2 -	5,88± 0,4	**22,0 ± 3,59 +**	**3,18±0,23 +**	1180±0,0 -	6,93±0,04 -	53,5±2,1 +	4,40±0,05 +
Adult	1115 ± 0,5	6,76 ± 0,4	>2411±0,5 -	6,82± 0,5 -	1432 ± 9,5 -	7,54±0,5 -	**16,1 ± 2,67 +**	**2,12±0,13 +**	1872±0,0 -	5,28±0,01 +	136,7±2,3 +	5,51±0,04 +
Crx	55,1 ± 4,1	5,18 ± 0,34	72,60 ± 4,33 -	5,49± 0,42	32,5 ± 3,7 +	4,53± 0,47	**7,36 ± 1,47 +**	**2,90±0,29 +**	83,4± 1,4 -	5,08 ± 0,10	16,5±1,8 +	3,91±0,34 +
Hepatitis	8,8 ± 2,5	**1,30 + 0,13**	13,30 ± 1,89 -	1,43± 0,20	6,2 ± 2,0	3,23±0,77 -	5,84 ± 1,09	2,82±0,29 -	10,9 ± 2,27	3,53±0,90 -	**3,9±1,0 +**	2,36±0,27 -
House	10,1 ± 3,6	2,97 ± 0,28	17,80 ± 6,46	3,52±0,24 -	8,6 ± 3,3	**1,85±0,51 +**	**3,48 ± 1,15 +**	1,85± 0,59	16,2 ± 3,99	3,70± 0,51	7,0 ± 1,5	2,38 ±0,32
Segment.	45,0 ± 4,8	2,33 ± 0,15	52,30 ± 3,95	2,49± 0,11	34,4 ± 2,8 +	**2,12 ± 0,21**	**21,96±2,11 +**	3,03± 0,18 -	74,5± 8,44 -	4,54±0,29 -	25,3±2,1 +	2,55 ± 0,13
Wave	354,1 ± 8,5	5,30 ± 0,31	397,9 ± 10,35 -	5,40± 0,29	252,3 ±6,6 +	5,59±0,46	**14,65± 2,37 +**	**1,65±0,23 +**	645,8±28,25 -	6,46±0,35 -	63,7±3,8 +	4,45±0,27 +
Splice	120 ± 8,5	2,60 ± 0,10	178,4 ± 5,78 -	2,81±0,09 -	51,3 ± 7,6 +	2,37±0,17 +	**14,60± 4,13 +**	2,77 ± 0,38	230,7±24,71 -	5,47±0,48 -	28,2±2,2 +	2,02±0,15 +
Covertype	787,6± 25,5	6,93 ± 0,09	1025,7±30,1 -	7,23±0,08 -	210,7± 38,72 +	6,55 ± 0,48	**24,07± 4,93 +**	**3,10±0,38 +**	1414,24±53,1-	6,87 ± 0,67	111,7±9,5 +	5,87±0,22 +
Letter	808,0 ± 0,4	3,95 ± 0,4	927,0 ± 1,0 -	4,24± 1,0	705,0± 9,0 +	4,37 ± 0,5	**136,5± 5,44 +**	**3,19±0,11 +**	1577,0±0,0 -	3,57±0,04 +	265,3±6,7 +	3,56±0,04 +
Nursery	318,6± 26,8	5,94 ± 0,05	609,3± 24,33 -	6,34±0,05 -	132,8± 16,23 +	5,34 ± 0,45	**25,04± 5,34 +**	**2,15±0,21 +**	433,7± 79,93 -	4,26±0,31 +	104,2±13,3+	5,22±0,52 +
Pendigits	181,0± 5,44	4,45 ± 0,03	211,6± 7,11 -	4,63±0,08 -	119,0± 5,96 +	4,21±0,15 +	**41,71± 4,07 +**	**3,36±0,15 +**	350,64±29,71-	5,02±0,11 -	82,33±3,5 +	3,94±0,14 +
CD-1	624,3± 27,6	4,96 ± 0,14	1383,6±38,5 -	5,18 ± 0,14	277,3± 48,78 +	4,76± 0,29	**16,9 ± 2,77 +**	**3,06±0,25 +**	797,4± 35,73 -	6,39±0,18 -	61,80±6,4 +	4,26±0,20 +
CD-2	598,9± 20,9	5,95 ± 0,13	1175,9±28,1 -	6,12 ± 0,09	215,5± 55,21 +	5,41±0,31 +	**17,31± 3,45 +**	**2,97±0,24 +**	667,1± 38,57 -	6,45±0,22 -	47,08±5,0 +	4,86±0,18 +
CD-3	386,1± 23,9	4,41 ± 0,09	922,5± 24,23 -	4,74±0,09 -	178,8± 42,84 +	4,52 ± 0,42	**17,25± 2,47 +**	**2,89±0,23 +**	499,5± 39,84 -	5,80±0,13 -	37,08±4,6 +	3,70±0,15 +
CD-4	46,1 ± 3,79	3,51 ± 0,36	411,0± 25,32 -	4,98±0,21 -	24,4 ± 7,79 +	3,33 ± 0,51	**10,18± 2,80 +**	**2,98±0,33 +**	57,90± 11,92	4,33±0,34 -	11,26±1,6 +	2,99±0,20 +
CD-5	221,0± 7,37	4,56 ± 0,19	737,6± 22,46 -	4,77 ± 0,19	124,8± 14,66 +	4,23 ± 0,27	**11,73± 2,82 +**	**2,83±0,35 +**	475,0± 20,91 -	5,51±0,30 -	28,98±3,3 +	3,74±0,17 +
CD-6	665,6± 55,4	5,10 ± 0,27	1424,7±25,36 -	5,56±0,14 -	296,4± 40,11 +	4,76 ± 0,28	**17,4 ± 2,72 +**	**2,97±0,25 +**	952,1± 46,71 -	6,38±0,28 -	70,46±7,1 +	4,51±0,26 +
CD-7	577,2± 37,0	5,88 ± 0,11	1237,1±29,06 -	6,11±0,10 -	272,5± 38,59 +	5,37 ± 0,32	**17,69± 1,83 +**	**3,00±0,33 +**	794,1± 11,85 -	6,49±0,17 -	50,19±5,0 +	4,82±0,20 +
CD-8	346,2± 35,5	4,19 ± 0,19	966,7± 27,13 -	4,79±0,15 -	183,6± 37,58 +	4,28 ± 0,27	**16,81± 2,57 +**	**2,88±0,26 +**	396,0 ± 39,98	5,87±0,48 -	33,28±4,8 +	3,47±0,15 +
CD-9	41,50± 7,60	3,48 ± 0,62	388,0± 19,26 -	4,98±0,28 -	30,7 ± 11,21	3,35 ± 0,42	**9,39± 2,77 +**	**2,94±0,38**	65,0 ± 11,4 -	4,58±0,37 -	11,68±1,9 +	3,01 ± 0,25
CD-10	209,2± 31,3	4,44 ± 0,43	753,3± 27,89 -	4,85 ± 0,25	140,4± 14,0 +	4,38 ± 0,51	**11,96± 2,67 +**	**2,93±0,33 +**	359,2± 23,54 -	6,92±0,65 -	21,84±4,5 +	3,57±0,38 +
N. de melhoras significat.			0	0	17	4	21	17	0	2	21	18
N. de pioras significat.			20	12	2	2	0	2	18	16	0	1

Da mesma forma que observado para os valores de $S = 3$, $S = 5$ e $S = 10$ (Tabelas 4.7, 4.8 e 4.9, respectivamente), para $S = 15$ (Tabela 4.10) os algoritmos C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS descobrem classificadores com um número consideravelmente menor de regras para todas as 22 bases de dados. Em 59 dos 66 casos a diferença entre o número de regras descobertas pelo C4.5 duplo, AG-Sozinho e pelo C4.5/AG-Grande-NS e o número de regras descobertas pelo C4.5 com poda é significativa.

A redução no número de regras do C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS em relação ao C4.5 com poda é superior a 40% em 12 das 22 bases de dados, é superior a 90% em 13 das 22 bases de dados e superior a 60% em 18 das 22 bases de dados, respectivamente.

A redução no número de regras geradas pelo AG-Grande-NS em relação ao C4.5 duplo é superior a 70% para 11 das 22 bases de dados. As maiores reduções ocorreram nas bases Connect e Adult, com redução de 91,7% e 90,5%, respectivamente, e a menor redução ocorreu na base House-votes (18%).

Sobre o quesito tamanho médio das regras descobertas, a Tabela 4.10 reforça a conclusão de que o AG-Sozinho obtém os menores tamanhos médio em 18 das 22 bases de dados. Em relação ao C4.5 com poda o C4.5 duplo obteve 4 e o AG-Grande-NS 18 melhoras significativas, sendo que pioras significativas ocorreram em uma base e duas bases, respectivamente.

Analisando a Figura 4.16 pode-se concluir que também para $S = 10$, as mesmas observações que foram feitas para $S = 3$, $S = 5$ e $S = 10$, quanto a cardinalidade o conjunto de regras descoberta pelos diversos algoritmos, se repetem, com exceção do algoritmo C4.5 para a base Connect onde ocorreu um pequeno aumento da cardinalidade do conjunto de regras descobertas em relação ao C4.5 com poda, ao invés de uma redução.

A partir da Figura 4.17 pode-se verificar que não só quanto ao número de regras, mas também no quesito tamanho médio das regras descobertas, o C4.5 sem poda descobre regras com tamanho médio maior que o tamanho médio das regras descobertas pelo C4.5 com poda para todas as bases. Esse aumento varia de $\cong 0.1\%$ a $\cong 43\%$. Em relação ao tamanho médio das regras descobertas pelos demais algoritmos em relação ao C4.5 com poda, pode-se concluir que:

- o C4.5 duplo reduz em 15 das 22 bases, sendo que essas reduções variam entre $\cong 1\%$ e $\cong 40\%$. Os aumentos variam entre $\cong 2\%$ e $\cong 150\%$;

• o C4.5/AG-Pequeno reduz em 5 das 22 bases (entre ≅ 1% e ≅ 30%) e os aumentos variam entre ≅ 8% e ≅ 110%;

• o C4.5/Ag-Grande-NS reduz em 20 das 22 bases (entre ≅ 10% e ≅ 25%) e os aumentos foram ≅ 10% (Segmentation) e ≅80% (Hepatitis); e

• o AG-Sozinho reduz em 19 das 22 bases (entre ≅ 15% e ≅ 70%) e os aumentos variam entre ≅ 7% e ≅115%.

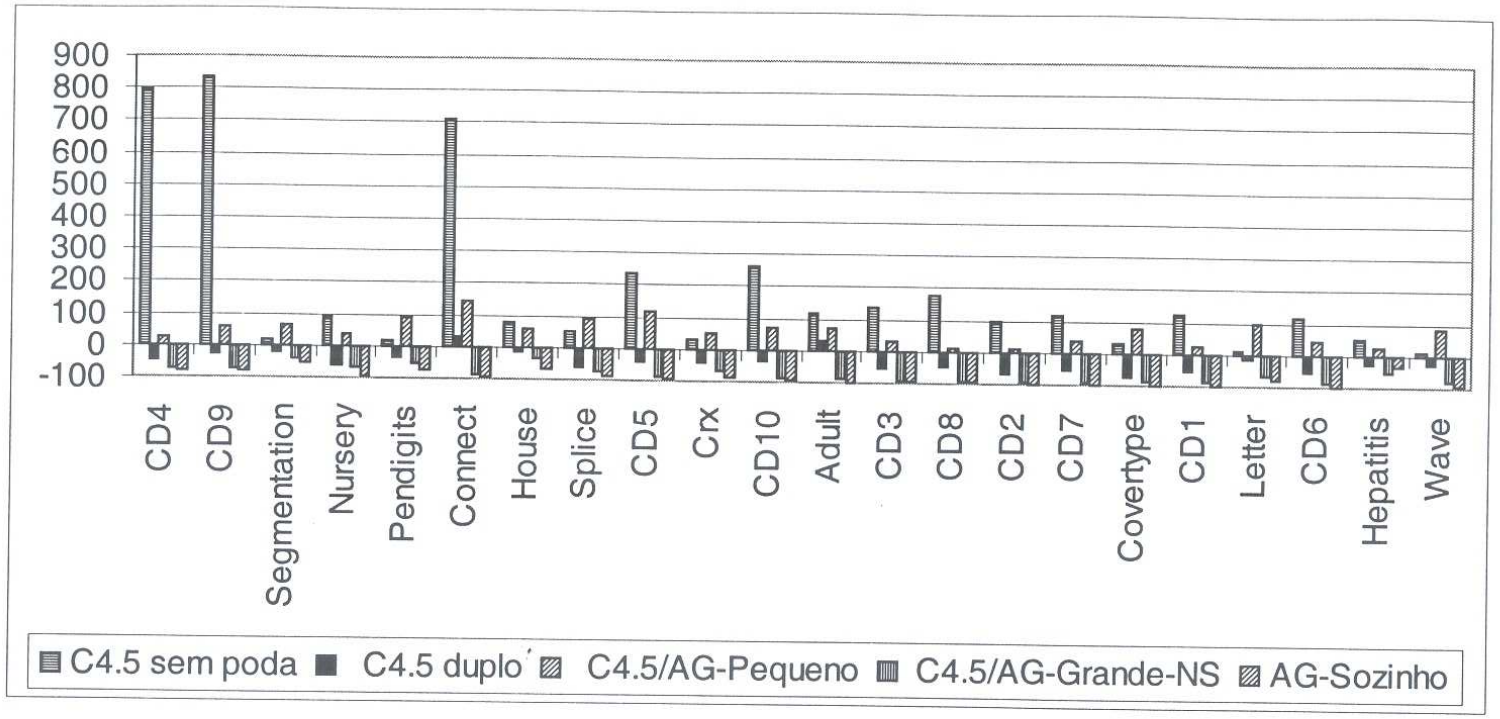


FIGURA 4.16 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO NÚMERO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA (S = 15)

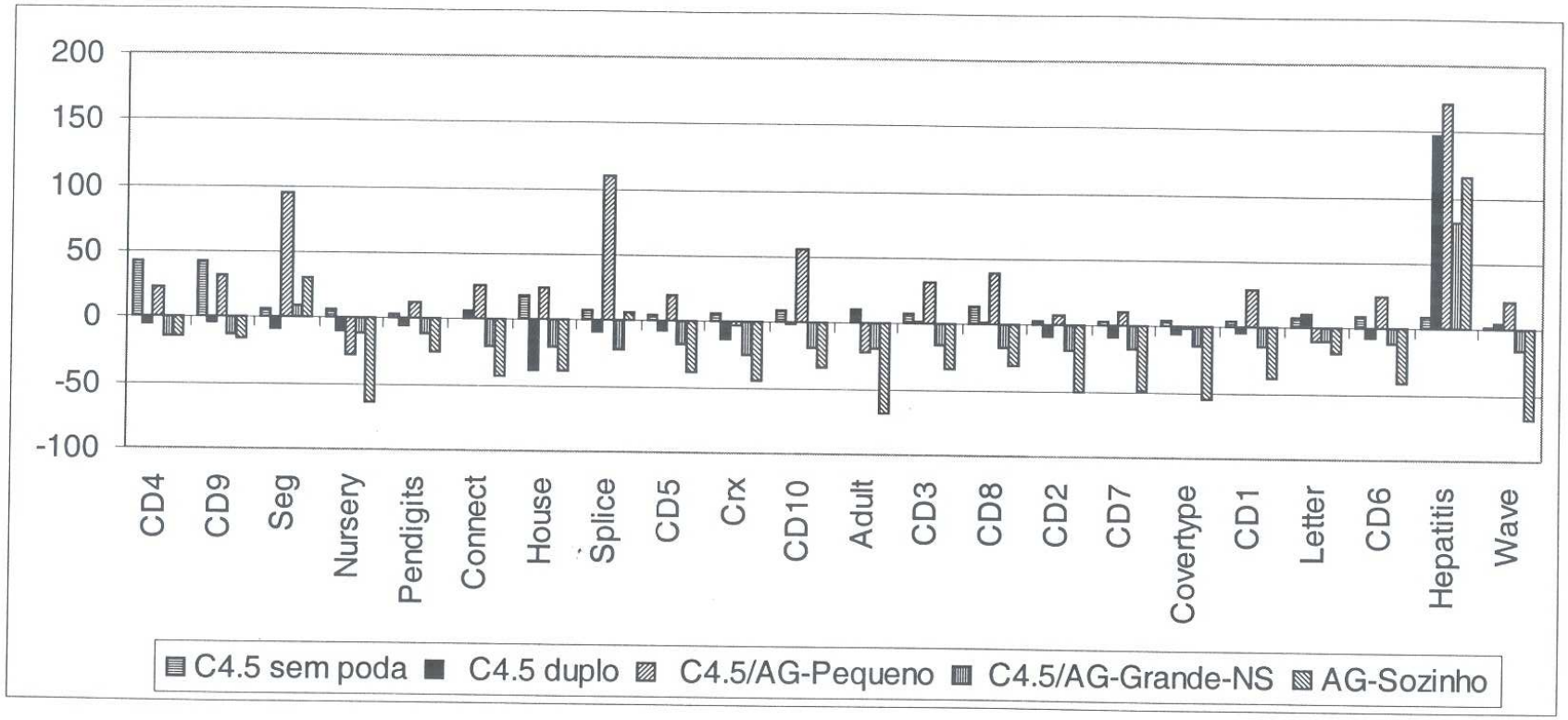


FIGURA 4.17 PORCENTAGEM DE AUMENTO/DIMINUIÇÃO DO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELOS ALGORITMOS TESTADOS EM RELAÇÃO AO TAMANHO MÉDIO DAS REGRAS DESCOBERTAS PELO C4.5 COM PODA ($S = 15$)

Em geral, dentre quase todos os métodos comparados, o C4.5-AG-Grande-NS obteve os melhores resultados tanto no número de regras quanto no tamanho médio das regras. O único método que obteve conjuntos de regras mais simples (menores) do que o C4.5/AG-Grande-NS foi o AG-Sozinho. Porém, conforme mencionado anteriormente, esse último não obteve bons resultados com relação à precisão preditiva, o que limita consideravelmente sua utilidade.

O C4.5/AG-Grande-NS não possui essa desvantagem. Ele obteve bons resultados em relação aos dois critérios: precisão preditiva e simplicidade.

4.7 COMENTÁRIOS SOBRE EFICIÊNCIA COMPUTACIONAL

Foram realizados experimentos comparando o tempo de processamento para todos os métodos usados nos experimentos aplicados à base de dados Connect, sendo que nesses experimentos os métodos foram executados na mesma máquina (um Pentium III com 192MB de RAM). A razão pela qual está sendo relatado o tempo de processamento para apenas uma base de dados se deve ao fato de que nas demais bases de dados os experimentos foram executados em diferentes máquinas, com distintas taxas de *clock* e capacidades de memória, ao mesmo tempo, para minimizar o tempo demandado para a realização dos experimentos. A base Connect foi escolhida por ser tratar da maior base utilizada para os experimentos, tendo 67557 exemplos.

Para a base Connect, o C4.5 com poda executou em 44 segundos. Este também é o tempo considerado para a versão C4.5 sem poda, dado que se trata da mesma execução. O C4.5 duplo executou em 52 segundos. O AG-Grande-NS executou em seis minutos (AG-Grande-NS) + 44 segundos (C4.5), o AG-Pequeno em 50 minutos (AG-Pequeno) + 44 segundos (C4.5) e o AG-Sozinho executou em 20 minutos. Estes tempos foram obtidos em experimentos realizados para o valor de $S = 15$. Embora o AG-Grande-NS use um conjunto de treinamento relativamente grande, sua execução é bem mais rápida que o AG-Pequeno, em virtude de este último precisar ser executado $d * c$ vezes, onde d é o número de pequenos disjuntos e c é o número de classes.

Embora o AG-Grande-NS seja mais lento que as três versões do C4.5, este acréscimo de tempo parece ser custo razoável, especialmente considerando que em aplicações do mundo real o tempo de execução de um algoritmo de *Data Mining* representa em geral apenas 10% ou 20% do tempo total gasto com o processo de descoberta de conhecimento

[MIC98]. Além disso, se necessário (se a base sendo minerada for realmente muito grande), o tempo de execução do AG-Grande-NS poderia ser bastante reduzido utilizando-se técnicas de processamento paralelo, já que AGs em geral podem ser paralelizados de forma bastante eficaz [FRE98].

4.8 EXPERIMENTOS COM A HEURÍSTICA DE PODA DO AG-GRANDE-NS

Para avaliar a nova heurística de poda proposta para o AG-Grande-NS, baseada na taxa de acerto de atributos (seção 3.2.5), realizou-se experimentos comparando a precisão preditiva de duas versões do AG-Grande-NS: (a) com essa heurística de poda; e (b) com a heurística de poda baseada no ganho de informação de condições da regra (pares de atributo-valor), descrita na seção 3.1.4. Cabe ressaltar que nesses experimentos todos os demais parâmetros do AG-Grande-NS foram mantidos iguais nas duas versões do algoritmo. Esses experimentos foram realizados sobre oito bases de dados, a saber: Adult, Connect, Crx, Hepatitis, House-votes, Segmentation, Wave e Splice.

Para facilitar o entendimento, foram adotas as seguintes identificações para estes experimentos:

- TxAc – Indica que a heurística de poda usada foi a taxa de acerto do atributo; e
- Gi – Indica que a heurística de poda usada foi o ganho de informação do atributo.

Estes dois experimentos adotaram o mesmo critério de obtenção dos resultados usado nas seções anteriores. Ou seja, eles foram realizados para os quatro valores de S, adotando o mesmo critério de 10 execuções dos AGs com variação da semente aleatória, usando validação cruzada (fator 10) para as bases Crx, Hepatitis, House-votes, Segmentation, Wave e Splice. Para as bases Adult e Connect foi adotada uma única partição de treinamento e de teste, apenas variando a semente aleatória em 10 execuções, no caso dos sistemas envolvendo AGs.

TABELA 4.11 TAXA DE ACERTO DO AG-GRANDE-NS VARIANDO HEURÍSTICA DE PODA DAS REGRAS – $S = 3$

Base Dados	TxAc	Gi
Connect	**77,86 ± 0,1**	77,81 ± 0,1
Adult	85,45 ± 0,1	**85,92 ± 0,1**
Crx	**93,69 ± 1,2**	93,09 ± 1,3
Hepatitis	**89,25 ± 9,5**	89,05 ± 9,5
House-votes	**97,18 ± 2,5**	97,15 ± 2,9
Segmentation	**81,56 ± 1,1**	81,45 ±1,1
Wave	83,86 ± 2,0	**83,75 ± 1,8**
Splice	**70,62 ± 8,6**	70,58 ± 9,0

As Tabelas 4.11, 4.12, 4.13 e 4.14 mostram os resultados obtidos nos experimentos realizados para comparar as duas heurísticas de poda para os valores de $S = 3$, $S = 5$, $S = 10$ e $S = 15$, respectivamente. Nestas tabelas a primeira coluna indica a base de dados, as duas próximas colunas apresentam os resultados para os classificadores híbridos, construídos a partir de variações do AG-Grande-NS: com poda baseada na taxa de acerto (TxAc) e com poda baseada no ganho de informação (Gi). Os experimentos que obtiveram o melhor resultado em relação aos demais estão em negrito. (Cabe ressaltar que a heurística de poda TxAc foi a heurística adotada nos experimentos realizados neste trabalho com o algoritmo AG-Grande-NS, cujos resultados foram mostrados anteriormente).

Vale salientar que a diferença entre as taxas obtidas pelas duas heurísticas não foi significativa em nenhum dos experimentos.

TABELA 4.12 TAXA DE ACERTO DO AG-GRANDE-NS VARIANDO HEURÍSTICA DE PODA DAS REGRAS – $S = 5$

Base Dados	TxAc	Gi
Connect	**77,85 ± 0,2**	77,66 ± 0,3
Adult	85,50 ± 0,2	**86,08 ± 0,1**
Crx	93,06 ± 1,6	**93,47 ± 1,9**
Hepatitis	89,48 ± 9,7	**89,71 ± 9,7**
House-votes	**97,44 ± 2,9**	97,39 ± 2,9
Segmentation	80,41 ± 1,0	**81,47 ± 0,9**
Wave	**85,37 ± 2,4**	85,13 2,4
Splice	**70,44 ± 7,8**	70,34 ± 7,6

TABELA 4.13 TAXA DE ACERTO DO AG-GRANDE-NS VARIANDO HEURÍSTICA DE PODA DAS REGRAS – $S = 10$

Base Dados	TxAc	Gi
Connect	**76,95 ± 0,1**	76,67 ± 0,1
Adult	80,04 ± 0,1	**81,08 ± 0,2**
Crx	91,66 ± 1,8	**91,86 ± 2,0**
Hepatitis	**95,05 ± 7,2**	93,95 ± 7,1
House-votes	**97,65 ± 2,0**	96,81 ± 1,8
Segmentation	78,68 ± 1,1	**78,81 ± 1,2**
Wave	83,95 ± 3,0	**83,97 ± 3,0**
Splice	**70,70 ± 6,3**	67,03 ± 4,2

Embora a diferença de *performance* (com respeito à precisão preditiva) entre as duas heurísticas de poda não seja significativa, a heurística baseada na taxa acerto (TxAc) tem a vantagem de ser de execução consideravelmente mais rápida. Foram realizados experimentos com a base Connect (a maior base de dados) para avaliação de tempo computacional, executando C4.5/AG-Grande-NS em uma mesma máquina duas vezes, uma vez para cada uma das duas heurísticas de poda. Conforme relatado na seção 4.7, C4.5/AG-

Grande-NS (TxAc) executou em seis minutos, enquanto que o C4.5/AG-Grande-NS (Gi) executou em 20 minutos.

Assim, na ausência de diferença significativa quanto à precisão preditiva, a maior eficiência computacional da heurística TxAc foi a razão pela qual essa heurística foi adotada como heurística de poda padrão do algoritmo AG-Grande-NS.

TABELA 4.14 TAXA DE ACERTO DO AG-GRANDE-NS VARIANDO HEURÍSTICA DE PODA DAS REGRAS - $S = 15$

Base Dados	TxAc	Gi
Connect	**76,01 ± 0,3**	75,99 ± 0,2
Adult	79,32 ± 0,2	**80,64 ± 0,2**
Crx	90,40 ± 2,4	**90,70 ± 2,6**
Hepatitis	**82,52 ± 7,0**	81,95 ± 23,9
House-votes	95,91 ± 2,3	**96,90 ± 1,9**
Segmentation	77,11 ± 1,9	**77,17 ± 1,7**
Wave	**82,65 ± 3,7**	82,62 ± 3,4
Splice	**70,62 ± 5,5**	66,57 ± 5,1

5 TRABALHOS RELACIONADOS

Neste capítulo serão discutidos trabalhos encontrados na literatura que têm como foco principal a questão do pequeno disjunto ou temas relacionados, no contexto de aprendizado de conceitos (*concept learning*) e *Data Mining*.

Liu et al. [LIU00b] apresentam uma nova técnica para organizar as regras descobertas em distintos níveis de detalhe. Os autores argumentam que o principal problema em análise de regras descobertas não decorre do fato dos algoritmos gerarem muitas regras, mas sim pela sua inabilidade em organizar e apresentar as regras descobertas de tal forma que seja fácil para o usuário analisá-las.

O algoritmo (GSE – *General rules summaries & exceptions*) consiste em várias fases. A primeira se preocupa em encontrar regras gerais, percorrendo uma árvore de decisão a partir do nó raiz para encontrar os nós folhas mais próximos da raiz que possam ser usados para formar regras significativas. Estas regras são chamadas de regras gerais de alto nível. A segunda fase consiste em encontrar exceções dessas regras gerais. A terceira fase consiste em encontrar as exceções das exceções, e assim por diante. É determinado se um nó da árvore deve formar ou não uma regra de exceção usando-se dois critérios: significância estatística e simplicidade.

Um padrão GSE consiste de três componentes, uma regra simples geral (se.. então), um sumário e um conjunto de exceções, na seguinte forma:

$X \rightarrow c_i$ (sup, conf)

Sumário

Exceção E_1, ..., E_n

onde

$X \rightarrow c_i$ é a regra geral, X é o conjunto de condições da regra, c_i é a classe prevista pela regra.
Sup e Conf são o suporte e a confiança da regra [AGR93], [SKI95].
Sumário disponibiliza algumas informações essenciais sobre a regra. Baseado no sumário, o usuário pode fazer a escolha de analisar apenas um determinado subconjunto de regras. O sumário não tem definição formal, ele varia de acordo com a necessidade da implementação.
$E = \{E_1, ..., E_n\}$ é o conjunto de exceções da regra geral. Cada E_j , j=1, ..., n , é um padrão GSE da mesma forma:

$X_j \rightarrow c_{jk}$ (sup,conf)

Sumário

Exceção E_{j1}, ... , E_{jw}

onde $(X_j \rightarrow c_{jk})$ é dita uma regra de exceção E_j, X_j é o conjunto de condições da regra de exceções E_j, e c_{jk} é a classe prevista pela regra de exceção E_j, sendo $c_{jk} \neq c_i$. Os exemplos que satisfizerem X_j devem satisfazer X.

Dado que o padrão GSE é hierárquico, ou seja, uma regra pode conter exceções que por sua vez também tem exceções e assim sucessivamente, o número de níveis desta hierarquia não deve ser grande, caso contrário dificultaria a compreensibilidade. No trabalho proposto o critério de parada desta estrutura hierárquica é determinado pelo usuário.

Algumas regras de exceção encontradas neste método podem ser consideradas pequenos disjuntos. Entretanto, os autores não tentam descobrir regras que cubram pequenos disjuntos com uma maior precisão preditiva. Este método foi proposto apenas como uma forma de sumarizar um grande conjunto de regras descobertas.

Ao contrário, o método proposto neste trabalho objetiva descobrir novas regras de pequenos disjuntos com maior poder preditivo que as regras descobertas por um algoritmo de árvore de decisão. A seguir são discutidos outros projetos mais diretamente relacionados ao objetivo deste trabalho.

Holte et al. [HOL89] investigaram três possíveis soluções para eliminar os pequenos disjuntos sem afetar a descoberta de "grandes" (não-pequenos) disjuntos, da seguinte forma:

(a) Eliminando todas as regras que cobrem um número de exemplos abaixo de um número limite previamente determinado;

(b) Eliminando apenas os pequenos disjuntos que apresentam uma baixa *performance* estimada. A *performance* é estimada pelo uso de um teste de significância estatística;

(c) Usando um *bias* de especificidade (isto é, favorecendo a descoberta de regras mais específicas) para pequenos disjuntos (sem alterar o *bias* de generalidade para os grandes disjuntos).

Com a eliminação de todas as regras que cobrem menos que determinado número de exemplos (solução (a)) existe o problema de não criar conceitos que caraterizem exemplos raros. Uma segunda objeção decorre do fato de que esta operação pode significativamente aumentar a taxa de erro do classificador, particularmente quando a base de dados tiver um grande número de pequenos disjuntos, o que de fato ocorre em várias bases de dados utilizadas nesta tese (conforme discutido na seção 4.4).

A eliminação dos pequenos disjuntos que apresentam baixa *performance* (solução (b)) já é adotada em vários algoritmos de classificação, como por exemplo CN2 [CLA89] [CLA91], CART [BRE84] e ID3 [QUI86]. Para eliminar um pequeno disjunto é necessário testar a significância estatística e medir a taxa de erro daquele disjunto. Isto porque para os pequenos disjuntos a taxa de erro não está relacionada à significância de forma simples. Também não está relacionada à entropia, uma medida que em geral é usada juntamente com o teste de significância.

Com relação à solução (c), os autores compararam os *biases* de especificidade máxima, especificidade seletiva[4] e de generalidade máxima. Para tal foi realizado um experimento utilizando um conjunto de treinamento com 200 exemplos obtidos de forma aleatória da base Kpa7KR (com dados sobre posições de jogo de xadrez) contendo 3196 exemplos. Os experimentos usaram duas definições de pequeno disjunto, $S = 5$ e $S = 9$. Lembre-se que S denota o número máximo de exemplos cobertos por uma regra para que ela seja considerada um pequeno disjunto).

Naturalmente, o uso de uma única base de dados limita a utilidade dos resultados computacionais obtidos nesse trabalho.

Os autores demonstram que existem sistemas de aprendizado que constróem de forma adequada conceitos para grandes disjuntos, mas que não tratam adequadamente os pequenos disjuntos. Entre as sugestões de trabalhos futuros está a construção de classificadores com distintos *biases* com o objetivo de tratar os grandes e pequenos disjuntos. O método híbrido proposto nesta tese pode ser considerado como seguindo esta linha de trabalho, sendo uma solução mais sofisticada para o problema de pequenos disjuntos do que as soluções discutidas em [HOL89]. Além disso, o método proposto neste trabalho foi avaliado em 22 bases de dados, enquanto o estudo de [HOL89] se concentrou em uma única base.

Quinlan [QUI91] realizou experimentos mostrando que só o tamanho do pequeno disjunto não é um parâmetro adequado para predizer a taxa de erro dos pequenos disjuntos. Disjuntos que predizem a classe da maioria são associados com menor taxa de erro do que aqueles que predizem a classe da minoria, comparando-se disjuntos de mesmo tamanho.

---

[4] *Bias* de especificidade seletiva ocorre quando um sistema de indução deve decidir pela criação ou não de um disjunto que cubra um determinado pequeno conjunto de exemplos de treinamento.

Para lidar com essa situação, o autor propôs uma modificação na estimativa de probabilidade dada pela fórmula de Bayes-Laplace. Nesse trabalho não foi proposto nenhum algoritmo novo para descoberta de regras de pequenos disjuntos.

Danyluk e Provost [DAN93] mostraram que em uma base de dados no domínio de telecomunicações pequenos disjuntos são necessários para uma alta precisão preditiva, ainda que os mesmos individualmente incorram em uma taxa de erro relativamente alta. A razão para isso é que, naquela base de dados, muitos exemplos pertencem a pequenos disjuntos. Portanto, o conjunto total de pequenos disjuntos é coletivamente muito importante, e as regras de pequenos disjuntos devem ser usadas para melhorar a precisão preditiva geral.

Os autores realizaram experimentos utilizando os algoritmos C4.5 [QUI93] e RL (algoritmo desenvolvido por Clearwater e Provost, sendo um algoritmo descendente do META-DENDRAL) e a base de dados utilizada foi MAX (NYMEX MAX) – um sistema especialista. Os autores mostram a variação do número de disjuntos aprendidos a partir da variação da definição de pequeno disjunto. A grande maioria dos disjuntos são pequenos. Eles comprovaram a afirmação de [HOL89] de que os pequenos disjuntos estão associados com um erro maior de classificação que os grandes disjuntos.

Nos experimentos com dados ruidosos, os algoritmos não foram capazes de obter uma boa taxa de acerto. Os autores concluíram que os problemas decorrem de dois motivos correlacionados:

- Da dificuldade em distinguir entre ruído e caso raro (exceção verdadeira). Na base MAX aproximadamente 50% dos exemplos são cobertos por pequenos disjuntos com $S = 10$;
- No domínio da aplicação em questão, erros de medida e classificação ocorrem com grande freqüência. Sendo assim, é difícil distinguir entre erros e casos raros. Nem o C4.5 nem o RL conseguiram taxas de acerto superiores a 60%.

Ting [TIN94] propôs o uso de um método híbrido de *Data Mining* para tratar dos pequenos disjuntos. Este método consiste em usar um algoritmo de árvore de decisão para tratar dos grandes disjuntos e um método de aprendizado baseado em instâncias (IBL - *Instance-Based Learning*) para tratar dos pequenos disjuntos. A idéia básica deste método híbrido é que os algoritmos IBL têm um *bias* de especificidade [AHA91], o qual é mais adequado para tratar de problemas de pequeno disjunto.

O classificador híbrido proposto nesta tese segue o mesmo princípio do método proposto por Ting, ou seja, os exemplos pertencentes a grandes disjuntos são classificados pelas regras oriundas do algoritmo de árvore de decisão (como por exemplo o algoritmo C4.5)

e os exemplos de pequenos disjuntos são classificados pelo IB1 [AHA91], um algoritmo simples do paradigma IBL, conforme discutido na seção 4.2.

Em seu trabalho Ting discute algumas formas de definir um pequeno disjunto, tais como:

a) tamanho absoluto de disjunto;

b) tamanho relativo de disjunto;

c) porcentagem de cobertura do conjunto de treinamento; e

d) taxa de erro do disjunto.

A primeira forma, tamanho absoluto de disjunto, foi utilizada nesta tese, conforme justificado na seção 4.3.

A segunda forma de definição, especificando um tamanho relativo de pequeno disjunto baseado em uma porcentagem do conjunto de treinamento, tenta resolver o problema causado por usar uma definição de tamanho fixo para situações onde o tamanho do conjunto de treinamento seja muito distinto entre diferentes bases de dados. Porém, essa definição de tamanho relativo apresenta alguns problemas, conforme foi discutido na seção 4.3.

A terceira forma requer que o total de cobertura de todos os pequenos disjuntos não exceda a uma porcentagem fixa do total de exemplos do conjunto de treinamento. Essa forma tem a desvantagem de ser relativamente complexa e pouco intuitiva, pois o fato de um determinado disjunto ser considerado como pequeno ou grande (não-pequeno) depende não apenas do disjunto em questão, mas também de outros disjuntos.

Na quarta forma, o uso da taxa de erro para definir um pequeno disjunto requer uma reorientação do significado de pequeno disjunto. Na verdade, substitui o conceito de pequeno disjunto pelo de *performance* ruim do disjunto, independente do seu tamanho. Assim, dentre dois disjuntos cobrindo o mesmo número de exemplos, um deles poderia ser considerado um pequeno disjunto e outro um grande (não-pequeno) disjunto, dependendo de suas estimativas de taxas de erro.

Ao comentar os resultados de seus experimentos o autor observa que o componente IBL do método híbrido, a saber o algoritmo IB1, tem dificuldade em trabalhar com bases de dados que contenham atributos irrelevantes.

Um fato que merece destaque é a indicação do autor, a partir do resultado de seus experimentos, que o *bias* de especificidade não é o único a ser aplicado no tratamento de pequenos disjuntos, sugerindo como trabalhos futuros o desenvolvimento de outros sistemas.

Conforme mostrado pelos experimentos computacionais descritos nesta tese, o método híbrido C4.5/IB1 proposto por Ting obteve bons resultados com relação à precisão preditiva, atingindo um nível de *performance* semelhante ao método híbrido C4.5/AG proposto nesta tese. Porém, cabe ressaltar que o método C4.5/IB1 não descobre regras compreensíveis generalizando os dados, enquanto o método C4.5/AG-Grande-NS levou à descoberta de conjuntos de regras bastante simples (compreensíveis).

O trabalho de Webb [WEB94] propõe um novo algoritmo para descoberta de uma lista ordenada de regras (denominada lista de decisão) que trabalha a partir da inserção de sucessivas regras no início da na lista em construção. Em contraste, o método clássico para construção de listas ordenadas trabalha adicionando sucessivas regras ao final da lista em construção.

O autor defende que este tipo de algoritmo constrói listas de decisões menores do que o método clássico. Um dos problemas que pode surgir nessa abordagem é atribuir uma maior importância aos pequenos disjuntos, ao contrário do método clássico. Desta forma é preciso implementar mecanismos que reduzam o impacto dos pequenos disjuntos. O primeiro mecanismo determina que antes da regra ser inserida na lista a sua cobertura (número de exemplos) satisfaça a determinado *threshold*. Um segundo mecanismo procura minimizar o impacto dos pequenos disjuntos pela troca de posição entre uma regra de pequeno disjunto e uma outra regra em uma posição posterior na lista de decisão, como forma de diminuir a taxa de erro de classificação da lista.

Os autores defendem, após realizarem experimentos sobre 12 bases de dados, que ambas as formas de tratar a questão dos pequenos disjuntos, poda e reposicionamento da regra na lista de decisão, aumentam a precisão preditiva da lista.

Naturalmente, o mecanismo para reordenação de regras proposto por Webb é específico para regras representadas na forma de listas de decisões. Essa é uma representação bem diferente da representação de regras adotada nesta tese, a qual consiste de um conjunto não-ordenado de regras.

Weiss [WEI95] investigou a interação de ruído com os exemplos raros (exceções verdadeiras), e mostrou que esta interação conduz a uma degradação na precisão preditiva, quando as regras de pequenos disjuntos são eliminadas. Entretanto, na prática estes resultados têm uma utilidade limitada, uma vez que a análise desta interação foi possível a partir do uso de bases de dados criadas artificialmente. Em bases de dados do mundo real os conceitos

corretos a serem descobertos não são conhecidos; desta forma não é possível fazer uma distinção clara entre o ruído e os exemplos raros.

Nos trabalhos de Van den Bosch et al. [BOS97], [DAE99], os autores defendem o uso de aprendizado baseado em instâncias para domínios onde a ocorrência de pequenos disjuntos seja significativa. Os autores estão especialmente interessados na tarefa de aprendizado de línguas, na qual, segundo os autores, ocorrem muitos pequenos disjuntos (ou seja, exceções). Em particular, eles focam o problema do aprendizado da pronúncia das palavras. Assim, esses trabalhos são mais relacionados à área de *text mining* do que à área de *Data Mining* propriamente dita. Portanto, os métodos usados nesses trabalhos estão sendo mencionados aqui apenas como exemplos da relevância do conceito de pequenos disjuntos, além do escopo definido nesta tese.

Lopes e Jorge [LOP00] apresentam uma metodologia para integração de regras e de casos. Aprendizado baseado em casos é usado quando uma regra não é suficiente para obter uma boa classificação. Inicialmente, todos os exemplos são usados para induzir um conjunto de regras com uma medida de qualidade satisfatória. Os exemplos que não são cobertos por estas regras são então tratados pelo método baseado em casos. Nesta abordagem, o processo é alternado entre aprendizado de regras e aprendizado baseado em casos. Se os exemplos iniciais puderem ser cobertos por regras de alta qualidade a abordagem baseada em casos não é acionada.

Cabe ressaltar que, em um alto nível de abstração, a idéia básica dessa metodologia é semelhante ao método híbrido árvore de decisão/IBL proposto por [TIN94]. Assim, essa metodologia também apresenta a desvantagem de que o componente de raciocínio baseado em casos do método híbrido não descobre regras compreensíveis generalizando os dados. Além disso, ao contrário do trabalho de Ting, o trabalho de Lopes e Jorge não tem foco no problema de pequenos disjuntos.

Weiss e Hirsh [WEI00] apresentam uma medida quantitativa para avaliar o efeito de pequenos disjuntos no processo de aprendizado. Os autores reportam experimentos com várias bases de dados para avaliar o impacto da presença dos pequenos disjuntos no processo de aprendizado, especialmente quanto a fatores tais como o uso ou não de uma estratégia de poda e níveis de ruído variados.

Os experimentos foram realizados com o algoritmo C4.5 [QUI93], dado o fato de tratar-se de um algoritmo bastante conhecido. Para a realização dos experimentos foram adotadas duas estratégias:

- execução do C4.5 com os parâmetros *default*, com poda; e
- execução do C4.5 com os parâmetros default, mas sem poda e desligando o critério de parada na construção da árvore (-m1). (A opção –m*x* interrompe o processo de criação de outros cortes durante o processo de construção da árvore, se o número de exemplos cobertos for inferior ao valor *x* especificado).

A realização dos experimentos envolveu sete execuções independentes de cada uma das versões do C4.5, computando-se a média desses resultados. Para cada execução foram selecionados 200 exemplos de forma aleatória para compor o conjunto de treinamento, enquanto os exemplos restantes constituíam o conjunto de teste. Foram utilizadas as bases de dados Kpa7KR (com dados sobre posições no jogo de xadrez) e Wisconsin *breats cancer*, sendo que ambas as bases contêm vários pequenos disjuntos. Porém, novamente o pequeno número de bases de dados (apenas duas bases) limita a validade dos resultados.

Em todo caso, os experimentos mostram que pequenos disjuntos têm um considerável impacto negativo na precisão preditiva obtida pelas duas versões do C4.5, em ambas as bases de dados.

Weiss [WEI98] também realizou experimentos mostrando que, quando ruído é adicionado a bases de dados do mundo real, os pequenos disjuntos contribuem de forma desproporcional e significativa para o número total de erros.

Os autores também discutem a questão da presença de ruído na base de treinamento e na base de teste. Primeiramente, a presença de ruído no conjunto de treinamento influencia o conceito a ser aprendido, já no conjunto de teste não. Dado que pequenos disjuntos são baseados no conceito aprendido, naturalmente pode-se concluir que ruído no conjunto de teste não pode gerar pequenos disjuntos. Além disso, ruído no conjunto de teste tende a afetar todos os disjuntos igualmente. Este fato explica o motivo pelo qual o efeito de ruído em pequenos disjuntos é menos dramático quando o ruído é aplicado tanto no conjunto de treinamento como no conjunto de teste, se comparado à situação do ruído estar limitado ao conjunto de treinamento. De certa forma ruído no conjunto de teste reduz a diferença relativa das taxas de erro entre os grandes e pequenos disjuntos. Quando o ruído é aplicado apenas no conjunto de teste, o efeito é fortemente diminuído, podendo inclusive desaparecer completamente se o

algoritmo tiver a habilidade de aprender o conceito correto apesar da introdução de ruído artificial.

Novamente, os resultados apresentados confirmam que a presença de pequenos disjuntos tem um impacto negativo na precisão preditiva em muitos casos. Entretanto, Weiss e seus colegas não propõem nenhuma solução nova para tratar a questão de pequenos disjuntos.

Carvalho e Freitas [CAR01] propuseram um algoritmo híbrido árvore de decisão / sistema imunológico para descobrir regras que tratam o problema do pequeno disjunto. A idéia básica é que os exemplos pertencentes a grandes disjuntos sejam classificados por regras produzidas por um algoritmo de árvore de decisão, enquanto exemplos pertencendo a pequenos disjuntos sejam classificados por regras produzidas por um algoritmo imunológico.

A principal característica abstraída do sistema imunológico natural, usada como inspiração para o projeto desse algoritmo imunológico, é o mecanismo de seleção clonal [HUN96], [DAS99], [CAS00a], [CAS00b]. Este mecanismo é usado para construir regras (anticorpos) que devem cobrir exemplos específicos (antígenos).

A idéia básica é que aprendizado no sistema imunológico significa aumentar o tamanho da população de anticorpos e aumentar a afinidade destes anticorpos em reconhecer qualquer antígeno. O algoritmo imunológico pode ser considerado como um tipo de algoritmo evolucionário, baseado em princípios da variação genética e da seleção natural. Entretanto, ele não inclui o operador de *crossover*, ele é baseado na seleção clonal e seu mecanismo de hipermutação.

Apesar desse trabalho propor um método alternativo para tratar o mesmo problema em foco nesta tese, ele não fez parte do escopo desta tese devido a dois fatores. Primeiro, esse trabalho ainda necessita de uma pesquisa mais aprofundada sobre os princípios do funcionamento do sistema imunológico natural (um assunto bastante complexo), pois a implementação do algoritmo correspondente acabou tendo muitas semelhanças com métodos evolucionários mais tradicionais. Segundo, os resultados dos experimentos realizados não foram suficientemente competitivos com o algoritmo C4.5 sozinho, usado como *baseline* nesta tese.

Em um alto nível de abstração, a idéia básica dos métodos híbridos propostos por Ting [TIN94], Lopes e Jorge [LOP00], e o método híbrido proposto neste trabalho são similares. Entretanto, os métodos propostos por aqueles autores têm a desvantagem que o correspondente algoritmo de aprendizado baseado em instâncias (ou casos) não descobre regras compreensíveis generalizando os dados. Em contraste, o método híbrido árvore de

decisão/AG, proposto nesta tese, descobre regras simples (compreensíveis) para cobrir os pequenos disjuntos, o que é uma característica importante no contexto de *Data Mining*.

Cabe ressaltar que nenhuma das soluções para o problema de pequenos disjuntos discutidas nesta seção envolve algoritmos genéticos.

6 RESUMO, CONCLUSÕES E TRABALHOS FUTUROS

6.1 RESUMO DAS CONTRIBUIÇÕES

A principal contribuição deste trabalho foi propor um método híbrido árvore de decisão/algoritmo genético (AG) para resolver o problema de pequenos disjuntos em *Data Mining*. A idéia básica desse método é que os exemplos pertencentes a grandes disjuntos são classificados por regras produzidas por um algoritmo de árvore de decisão, enquanto os exemplos pertencentes a pequenos disjuntos são classificados por um AG.

Até onde se estende o conhecimento da autora, esse método híbrido, por si só, representa uma abordagem nova para resolver aquele problema. É importante salientar que o problema de pequenos disjuntos é um problema relativamente pouco investigado na literatura, conforme discutido no Capítulo 5.

Além disso, este trabalho também propôs dois AGs novos, especificamente projetados para descobrir regras de pequenos disjuntos. Esses dois AGs foram denominados de AG-Pequeno e AG-Grande-NS, em virtude de usarem conjuntos de treinamento pequenos e grandes, respectivamente, conforme explicado anteriormente. (Cabe lembrar que o termo "NS" na segunda versão do AG refere-se à técnica de *Niching* Seqüencial). Os dois AGs foram usados para instanciar o componente do AG do método híbrido, levando a duas versões daquele método, denominadas C4.5/AG-Pequeno e C4.5/AG-Grande-NS. Em ambas versões do método híbrido o algoritmo C4.5 foi utilizado para instanciar o componente de árvore de decisão, mas a princípio outros algoritmos de árvore de decisão poderiam ser utilizados, alternativamente. O C4.5 foi escolhido por ser um algoritmo muito conhecido e freqüentemente usado como "padrão de comparação" (*baseline*) em relação a outros algoritmos.

Outra contribuição deste trabalho, em termos de algoritmos, foi a implementação do C4.5 duplo (conforme explicado na seção 4.2), que pode, até certo ponto, ser considerado um novo algoritmo para resolver o problema de pequenos disjuntos. O C4.5 duplo, mais precisamente, consiste em uma nova forma de usar o C4.5, em vez de um algoritmo novo propriamente dito. Até onde se estende o conhecimento da autora, o uso do C4.5 duplo para resolver o problema de pequenos disjuntos ainda não foi relatado na literatura. Em todo caso, cabe ressaltar que a idéia básica do C4.5 duplo não foi proposta por esta autora, mas sim por

um revisor anônimo de um artigo submetido a uma conferência, a quem esta autora gostaria de agradecer.

Finalmente, outra contribuição deste trabalho foi mostrar que a proporção de exemplos pertencentes a pequenos disjuntos é maior do que se imaginava inicialmente em várias bases de dados. Isso reforça o argumento de que o problema de pequenos disjuntos é um problema importante em *Data Mining* (conforme discutido na Introdução).

6.2 RESUMO DOS RESULTADOS COMPUTACIONAIS

As duas versões do método híbrido proposto neste trabalho, a saber C4.5/AG-Pequeno e C4.5/AG-Grande-NS, foram comparadas com três versões do C4.5 (C4.5 com poda (versão *default* do C4.5), C4.5 sem poda e C4.5 duplo), com o AG-Sozinho e com o C4.5/IB1 em 22 bases de dados.

Experimentos foram realizados com quatro definições diferentes de pequeno disjunto, variando-se o valor do parâmetro S. Esse parâmetro especifica o número máximo de exemplos pertencentes a um nó folha da árvore de decisão para que o nó em questão seja considerado um pequeno disjunto. Os quatro valores de S utilizados nos experimentos foram $S = 3$, $S = 5$, $S = 10$ e $S = 15$.

Para valores de $S = 10$ e $S = 15$ os sete algoritmos mencionados anteriormente foram comparados. Para valores de $S = 3$ e $S = 5$, seis daqueles sete algoritmos foram comparados. O algoritmo não utilizado para $S = 3$ e $S = 5$ foi o C4.5/AG-Pequeno. A razão para isso é que, conforme mencionado na seção 3.2.1, o C4.5/AG-Pequeno intuitivamente não faz muito sentido para $S = 3$ e $S = 5$, já que nesses casos haveriam muito poucos exemplos de treinamento para o AG-Pequeno.

Os algoritmos foram avaliados principalmente de acordo com dois critérios, a saber, a precisão preditiva e a simplicidade do conjunto de regras descobertas em cada algoritmo. A precisão preditiva foi medida pela taxa de acerto em dados de teste (separados dos dados de treinamento). A simplicidade foi medida pelo número de regras descobertas e pelo número médio de condições por regras.

Os resultados com respeito a esse dois critérios são sumarizados nas duas próximas seções, respectivamente. Cabe ressaltar que esses resultados foram obtidos realizando-se um número bastante grande de experimentos. Para a obtenção destes resultados, ao todo, considerando-se todas as iterações de validação cruzada, variações de sementes aleatórias,

diferentes valores de S e todas as bases de dados, o algoritmo C4.5 foi executado 193 vezes; a segunda execução do C4.5 no algoritmo C4.5 duplo foi realizada 772 vezes; o algoritmo AG-Sozinho foi executado 1930 vezes; o algoritmo IB1 foi executado 772 vezes; o algoritmo AG-Pequeno foi executado 3860 vezes; e o algoritmo AG-Grande-NS foi executado 7720 vezes.

6.2.1 Resultados com Relação à Precisão Preditiva

A Tabela 6.1 apresenta um sumário dos resultados obtidos nas 22 bases de dados. Nessa análise, considera-se o algoritmo C4.5 com poda (versão *default*) como um padrão de comparação (ou *baseline*). Assim, a tabela mostra, para cada um dos outros algoritmos, o quão melhor ou pior foi o seu desempenho em relação ao C4.5 com poda. Mais precisamente cada célula *i, j* da tabela contém um par de números na forma X / Y, onde X (Y) representa o número de bases de dados (dentre as 22 usadas nos experimentos) nas quais a taxa de acerto do algoritmo identificado na *j-ésima* coluna foi significativamente superior (inferior) à taxa de acerto do C4.5 com poda, para o valor de S identificado na *i-ésima* linha. Portanto, quanto maior o valor de X e quanto menor o valor de Y, melhor foi o desempenho do correspondente algoritmo.

TABELA 6.1 SUMÁRIO DA COMPARAÇÃO DA TAXA DE ACERTO

Tamanho do Pequeno Disjunto (S)	C4.5 sem poda	C4.5 duplo	AG-Sozinho	C4.5/IB1	C4.5/AG-Pequeno	C4.5/AG-Grande-NS
3	0 / 7	8 / 4	1 / 11	8 / 1	-	9 / 2
5	0 / 7	6 / 3	1 / 11	8 / 1	-	8 / 2
10	0 / 7	4 / 4	1 / 11	8 / 1	7 / 4	9 / 2
15	0 / 7	7 / 4	1 / 11	10 / 1	7 / 3	8 / 3

Em relação à precisão preditiva (Tabela 6.1), pode-se concluir que o C4.5 sem poda e o AG-Sozinho obtiveram os piores resultados. O resultado ruim do C4.5 sem poda parece ser devido ao fato da árvore não podada estar demasiadamente ajustada (*overfitted*) aos dados. O resultado ruim obtido pelo AG-Sozinho pode ser justificado pelo fato desse algoritmo ser na verdade o AG-Grande-NS aplicado a todo o conjunto de treinamento, sem distinguir entre pequenos e grande disjuntos. (Conforme explicado na seção 4.2). O algoritmo AG-Grande-NS não foi desenvolvido para ser usado desse modo. Ele foi desenvolvido para classificar apenas pequenos disjuntos.

O C4.5 duplo obteve melhores resultados em relação ao C4.5 com poda, mas mesmo assim piores resultados se comparado com os resultados obtidos pelos algoritmos C4.5/IB1, C4.5/AG-Pequeno e C4.5/AG-Grande-NS. O algoritmo C4.5/AG-Pequeno não foi avaliado para os valores de $S = 3$ e $S = 5$ (conforme explicado anteriormente). Para os valores de $S = 10$ e $S = 15$ esse algoritmo obteve melhores resultados em relação ao C4.5 com poda e um pouco melhores que os resultados obtidos pelo C4.5 duplo, porém inferior aos resultados obtidos pelos algoritmos C4.5/IB1 e C4.5/AG-Grande-NS.

Já os algoritmos C4.5/IB1 e C4.5/AG-Grande-NS obtiveram os melhores resultados não apenas em relação ao C4.5 com poda, mas também em relação aos demais algoritmos considerados na avaliação.

6.2.2 Resultados com relação à Simplicidade

Da mesma forma que a Tabela 6.1, as Tabelas 6.2 e 6.3 apresentam um sumário dos resultados obtidos nas 22 bases de dados, agora em relação à simplicidade do conjunto de regras descobertas, conforme os critérios: o número de regras (Tabela 6.2) e o número médio de condições (tamanho) por regra (Tabela 6.3) do conjunto de regras descobertas. Nessa análise, também se considera o algoritmo C4.5 com poda como um padrão de comparação (ou *baseline*). Desta forma, as tabelas mostram, para cada um dos outros algoritmos, o quão melhor/pior foi o seu desempenho em relação ao C4.5 com poda. O critério de apresentação é o mesmo adotado para a Tabela 6.1.

Naturalmente, não são apresentados resultados de simplicidade de regras para o algoritmo C4.5/IB1, pois o mesmo não descobre conhecimento representado na forma de regras.

Em relação ao número de regras descobertas (Tabela 6.2), pode-se concluir que o C4.5 sem poda e o C4.5/AG-Pequeno obtiveram os piores resultados. Em geral, em torno de 20 das 22 bases de dados, mas esses algoritmos descobriram um conjunto de regras significativamente maior (pior resultado) do que o C4.5 com poda. O C4.5 duplo obteve resultados significativamente melhores (menos regras) em relação ao C4.5 com poda na grande maioria das bases de dados, obteve resultados um pouco piores do que os resultados obtidos pelos algoritmos AG-Sozinho e C4.5/AG-Grande-NS. Note que em geral esses dois últimos algoritmos obtiveram resultados significativamente melhores que o C4.5 com poda

em 21 ou 20 bases de dados, sem obter resultados significativamente piores em nenhuma base.

TABELA 6.2 SUMÁRIO DA COMPARAÇÃO DO NÚMERO DE REGRAS DESCOBERTAS

Tamanho do Pequeno Disjunto (*S*)	C4.5 sem poda	C4.5 duplo	AG-Sozinho	C4.5/AG-Pequeno	C4.5/AG-Grande-NS
3	0 / 20	19 / 0	21 / 0	-	20 / 0
5	0 / 20	19 / 0	21 / 0	-	20 / 0
10	0 / 20	18 / 2	21 / 0	0 / 20	21 / 0
15	0 / 20	17 / 2	21 / 0	0 / 18	21 / 0

Em relação ao número médio de condições (tamanho) por regra (Tabela 6.3), pode-se concluir que o C4.5 sem poda obteve os piores resultados. O C4.5/AG-Pequeno obteve resultados um pouco melhores em relação ao C4.5 sem poda, mas mesmo assim piores se comparado com os resultados obtidos pelos algoritmos C4.5 duplo, AG-Sozinho e C4.5/AG-Grande-NS. O algoritmo C4.5 duplo obteve melhores resultados em relação ao C4.5 com poda e o C4.5/AG-Pequeno, porém não superou os resultados obtidos pelos algoritmos AG-Sozinho e C4.5/AG-Grande-NS.

TABELA 6.3 SUMÁRIO DA COMPARAÇÃO DO NÚMERO MÉDIO DE CONDIÇÕES POR REGRA DESCOBERTA

Tamanho do Pequeno Disjunto (*S*)	C4.5 sem poda	C4.5 duplo	AG-Sozinho	C4.5/AG-Pequeno	C4.5/AG-Grande-NS
3	0 / 12	4 / 0	17 / 2	-	2 / 0
5	0 / 12	5 / 2	17 / 2	-	6 / 1
10	0 / 12	8 / 1	17 / 2	3 / 15	16 / 1
15	0 / 12	4 / 2	17 / 2	2 / 6	18 / 1

Já o algoritmo C4.5/AG-Grande-NS obteve melhores resultados (em relação tanto ao número de regras quanto ao número médio de condições por regra) não apenas em relação ao C4.5 com poda, mas também em relação aos demais algoritmos considerados na avaliação, com exceção do AG-Sozinho.

Vale lembrar que apesar do AG-Sozinho obter bons resultados quanto à simplicidade do conjunto de regras descobertas, ele não obteve bons resultados quanto a precisão preditiva, o que não o coloca entre os algoritmos mais competitivos para a descoberta de regras em bases com alta incidência de pequenos disjuntos.

6.2.3 Conclusão Geral sobre os Resultados Computacionais

Os resultados computacionais obtidos a partir dos diversos experimentos realizados nesta tese constituem evidência significativa que, com relação à precisão preditiva, os algoritmos C4.5 duplo, o C4.5/IB1 e o C4.5/AG-Grande-NS são algoritmos competitivos como alternativa de solução para a construção de classificadores para bases de dados com alta incidência de pequenos disjuntos.

Entre estes quatro algoritmos é possível destacar os algoritmos C4.5/IB1 e o C4.5/AG-Grande-NS como sendo os algoritmos que melhor satisfizeram o quesito de maximização da precisão preditiva do classificador gerado.

Porém, a questão fundamental é que, dentre esses quatro algoritmos, apenas o algoritmo C4.5/AG-Grande-NS descobre conhecimento altamente compreensível (simples). Portanto, em aplicações onde se deseja maximizar tanto a precisão preditiva quanto a simplicidade, a melhor alternativa seria a adoção do C4.5/AG-Grande-NS para construir classificadores no caso de bases com presença significativa de pequenos disjuntos.

O algoritmo C4.5 sem poda não se apresentou competitivo tanto na avaliação da precisão preditiva, quanto na avaliação da simplicidade. O algoritmo AG-Sozinho só seria considerado como uma alternativa de solução caso a prioridade determinada pelo usuário fosse o quesito simplicidade, dado que foi o algoritmo com melhor *performance* neste quesito. Porém, como em geral o usuário está interessado não apenas na questão do fácil entendimento do conhecimento descoberto, mas também na alta precisão preditiva do classificador, este algoritmo fica com uma indicação bastante restrita.

Os resultados também constituem evidência para a afirmação de [ANA98] de que os sistemas híbridos têm a vantagem de superar as limitações que cada um de seus componentes apresentam quando utilizados de forma isolada. Essa afirmação pode ser verificada ao se analisar os resultados do C4.5 com poda e o AG-Sozinho. Enquanto o C4.5 com poda privilegia a taxa de acerto dos classificadores gerados, o AG-Sozinho obtém classificadores mais simples. Porém, sua precisão preditiva não é competitiva com a do algoritmo C4.5/AG-Grande-NS.

É importante ressaltar que:

- Esta tese propôs um método bastante competitivo com alguns métodos descritos na literatura para tratar o problema do pequeno disjunto;

• Até onde a pesquisa na literatura pode comprovar, trata-se de método original, o qual foi avaliado através de um grande número de experimentos, fazendo com que os resultados computacionais apresentados sejam relevantes.

6.3 TRABALHOS FUTUROS

Existem várias direções de pesquisa que poderiam complementar alguns dos resultados / conclusões obtidos nesta tese. Em primeiro lugar, poderia ser realizado um estudo de *metalearning* [HOR94] sobre os resultados obtidos com os experimentos realizados com as 22 bases de dados, tentando identificar em qual situação (dependendo das características da base de dados a ser minerada) cada um dos algoritmos usados nos experimentos seria o mais indicado. Por exemplo, seria possível construir um classificador sobre uma base de treinamento que contivesse 88 exemplos (22 bases de dados * 4 valores de S). Para cada exemplo, a classe seria o nome do algoritmo que obteve a melhor taxa de acerto para a base de dados e o valor de S associados com o exemplo. A questão, mais importante a ser pesquisada é quais seriam os atributos previsores que iriam compor essa base.

Vale ressaltar que foram realizados alguns experimentos preliminares com este objetivo, selecionando alguns atributos como previsores - número de exemplos da base, número de pequenos disjuntos, porcentagem de exemplos pertencentes a pequenos disjuntos, etc. - porém, não se chegou a uma conclusão satisfatória. Esse resultado não satisfatório decorreu do fato de não se tratar de um problema trivial, ou seja, merece um estudo muito mais detalhado.

Outra direção de pesquisa que pode derivar deste trabalho consiste em otimizar os algoritmos que compõem o método híbrido. Por exemplo, pode-se adotar outros algoritmos de árvore de decisão, ou mesmo, otimizar o valor de vários parâmetros do AG tais como o tamanho da população, o número de gerações, etc.

Ainda sobre a questão de otimização do AG, outra direção de pesquisa seria adotar uma função de avaliação (*fitness*) que levasse em consideração não apenas a questão da precisão preditiva, mas também a questão da simplicidade. Isso poderia ser realizado de pelo menos duas formas, mutuamente exclusivas entre si. A primeira seria usar uma função de avaliação que fosse uma soma ponderada de valores de precisão preditiva e simplicidade. Essa abordagem introduz o difícil problema de otimizar os pesos atribuídos aos quesitos de precisão preditiva e simplicidade.

Alternativamente, poderia ser desenvolvido um AG com uma função de avaliação multiobjetiva [DEB01]. Isso evita o problema de atribuição de pesos aos dois quesitos, mas aumenta significativamente a complexidade do AG.

Outra pesquisa poderia aprofundar a análise da relação existente entre a precisão preditiva e a simplicidade das regras descobertas. Para tal, as regras descobertas seriam avaliadas com o objetivo de verificar qual a precisão preditiva de regras de igual tamanho descobertas pelos algoritmos C4.5 com poda, C4.5 duplo, AG-Sozinho, C4.5/AG-Pequeno e C4.5/AG-Grande-NS. Poder-se-ia também medir qual o percentual de erros de classificação que é devido a regras de pequenos disjuntos e de grandes disjuntos, separadamente, no caso dos métodos híbridos.

Finalmente, outra direção de pesquisa consiste em estender os experimentos computacionais para incluir os resultados do algoritmo IB1 (ou outro algoritmo baseado em instância mais sofisticado) sozinho; ou seja, aplicado a todo o conjunto de treinamento original. Assim, seria possível avaliar a eficácia do algoritmo híbrido C4.5/IB1 em relação a seus dois algoritmos componentes usados separadamente.

REFERÊNCIAS BIBLIOGRÁFICAS

[AGR93] AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining Associations between Sets of Items in Massive Databases. *Proc. of the ACM-SIGMOD 1993 Int'l Conference on Management of Data*, Washington D.C., May 1993, p.207-216.

[AHA91] AHA, D.W.; KIBLER, D.; ALBERT, M.K. Instance-based learning algorithms. *Machine Learning*, 6, 1991, p.37-66.

[AHA98] AHA, D. Feature weighting for lazy learning algorithms. In H.LIU and H.Motoda, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Norwell MA: Kuwer, 1998.

[ADR96] ADRIAANS, P.; ZABTINGE, D. *Data Mining*, England, Addison Wesley Longman. 1996.

[ANA98] ANAND, S.S.; HUGHES, J.G. Hybrid Data Mining Systems: The Next Generation. (Eds.): Research and Development in Knowledge Discovery and Data Mining, *Second Pacific-Asia Conference, PAKDD-98*, Melbourne, Australia, April 15-17, 1998, Proceedings.Lecture Notes in Computer Science, Vol. 1394, Springer, 1998, p. 13 - 24.

[ATK97] ATKESON, C.; MOORE, A.; SCHAAL, S. Locally Wighted Learning. In: AHA, D. (Ed.), *Lazy Learning*, Netherlands: Kluwer Academic Publishers. 1997, p.11-73.

[BAC91] BACK, T.; HOFFMEISTER, F. Extended selection mechanisms in genetic algorithms. *Proc. of IV Int. Conf. on Genetic Algorithms - ICGA*, San Diego, USA. 1991, p.92-99.

[BAC94] BACK, T.Selective Pressure in Evolutionary Algoruths: A Characterization of Selection Mechanisms. *Proc. Of the First IEEE Conf. On Evolutionary Computation*. IEEE World Congress on Computational Intelligence. 1994. p.57-62.

[BEA93] BEASLEY, D.; BULL, D.R. M.; RALPH, R.A. Sequential Niche Technique for Multimodal Function Optimization. *Evolutionary Computation 1(2)*, MIT Press. 1993, p.101-125.

[BER97a] BERSON, A.; SMITH, S.J. *Data Warehousing, Data Mining, and OLAP*, USA, McGraw-Hill. 1997.

[BER97b] BERRY, M.J.A.; LINOFF, G. *Data Mining Techniques: for marketing, sales, and customer support*, USA, John Wiley &Sons, Inc. 1997.

[BLI95] BLICKLE, T.; THIELE, L. A Comparison of Slection Schemes used in Genetic Algorithms. *TIK-Report*, Swiss Federal Institute of Technology. Zurich. 1995.

[BLI00] BLICKLE, T. Tournament selection, in Back, T, Fogel, D.B., Michalewicz (Eds.), *Evolutionary Computation 1*, Philadelphia: Institute of Physics Publishing. 2000, p.181-186.

[BOO00] BOOKER, L.B.; FOGEL, D.B.; WHITLEY, D.; ANGELINE, P.J.; EIBEN, A.E. Recombination, in Back, T, Fogel, D.B., Michalewicz (Eds.), *Evolutionary Computation 1*, Philadelphia: Institute of Physics Publishing. 2000, p.256-307.

[BOS97] BOSCH, V. ;WEIJTERS, A.; HERICK, V.; DAELEMANS, H.J. When Small Disjuncts Abound, Try Lazy Learning: A Case Study.In *Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning*, 1997, p. 109-118.

[BRE84] BREIMAN, L.; FRIEDMAN, J.H.; OLSHEN, R.A.; STONE, C.J. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, Ca. 1984.

[BRE97] BRESLOW, L.A.; AHA, D.W. Simplifying Decision Trees: A Survey.*The Knowledge Engineering Review*, 12(10). March, 1997, p.1-40.

[CAR00a] CARVALHO, D.R; FREITAS, A.A. A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in Data Mining. *Proc. 2000 Genetic and Evolutionary Computation Conf. (GECCO-2000),* Las Vegas, NV, USA. July, 2000, p.1061-1068.

[CAR00b] CARVALHO, D.R.; FREITAS, A.A. A genetic algorithm-based solution for the problem of small disjuncts. Principles of Data Mining and Knowledge Discovery (*Proc. 4th European Conf., PKDD-2000*. Lyon, France). Lecture Notes in Artificial Intelligence 1910, Springer-Verlag. 2000, p.345-352.

[CAR01] CARVALHO,D.R.; FREITAS, A.A. An immunological algorithm for discovering small-disjunct rules in data mining. *Proc. Graduate Student Workshop at GECCO-2001*, San Francisco, CA, USA. July 2001, p. 401-404.

[CAR02a] CARVALHO, D.R; FREITAS, A.A. A genetic algorithm with sequential niching for discovering small-disjunct rules. A ser publicado nos *Proc. 2002 Genetic and Evolutionary Computation Conf. (GECCO-2002),* New York. July, 2002.

[CAR02b] CARVALHO, D.R; FREITAS, A.A. A genetic algorithm for discovering small-disjunct rules in data mining. Aceito para publicação no *Applied Soft Computing (ASC), The official journal of World Federation of Soft Computing (WFSC).* 2002.

[CAS00a] CASTRO, L.N; ZUBEN, F.J. 2000. An Evolutionary Immune Network for Data Clustering, *Proc. SBRN 2000, Brazilian Symposium on Artificial Neural Networks*, Rio de Janeiro, Brazil. 2000, p 84-89.

[CAS00b] CASTRO, L.N; ZUBEN, F.J. 2000. The Clonal Selection Algorithm with Engineering Applications, In: Wu, A. S. (Ed.) *Proc of the 2000 Genetic and Evolutionary Computation Conference. (GECCO-2000*), Workshop Program – Workshop Immune Systems. Las Vegas, NV, USA. July 2000, p. 36-37.

[CHE96] CHEN, M.; HAN, J.; YU, P.S. Data Mining: An Overview from Database Perspective, *IEEE Transactions on Knowledge and Data Engineering,* 8(6). December, 1996, p.866-883.

[CLA89] CLARK, P.; NIBLET, T. The CN2 Induction Algorithm. *Machine Learning* 3(4). Netherlands: Kluwer. 1989, p. 261-283.

[CLA91] CLARK, P.; BOSWELL, R. Rule induction with CN2: Some recent improvements. In Y. Kodratoff, (Ed.), *Machine Learning - EWSL-91*, Berlin: Springer-Verlag. 1991, p.151-163.

[CON95] CONGDON, C.B. *A Comparison of Genetic Algorithms and other Machine Learning Systems on a Complex Classification Task from Common Disease Resarch.* Ph.D. Thesis in Computer Science in the University of Michigan. 1995, 168p.

[COV91] COVER, T.M.; THOMAS, J.A. *Elements of Information Theory*. New York: John Wiley & Sons, Inc. 1991.

[DAE99] DAELEMANS, W.; BOSCH, A.V.; ZAVREL, J. Forgetting Exceptions is Harmful in Language Learning. Kuwer Academic Publishers. Netherlands. 1999, p.1-34.

[DAN93] DANYLUK, A.P.; PROVOST, F. Small Disjuncts in Action: Learning to Diagnose Errors in the Local Loop of the Telephone Network, *Proc. 10th International Conf. Machine Learning*, San Francisco, CA: Morgan Kaufmann. 1993, p.81-88.

[DAS99] DASGUPTA, D. *Artificial Immune Systems and Their Applications. Springer-*Verlag, 1999.

[DEB00] DEB, K. Introduction to Selection, in Back, T, Fogel, D.B., Michalewicz (Eds.), *Evolutionary Computation 1,* Philadelphia: Institute of Physics Publishing. 2000, p.166-171.

[DEB01] DEB, K. Multi-Objective Optimization using Evolutionary Algorithms. New York. John Wiley & Sons. 2001.

[DEJ93] DE JONG, K.A.; SPEARS, W.M.; GORDON, D.F. Using genetic algorithms for concept learning. *Machine Learning, 13.* 1993, p.161–188.

[DEC95] DECKER, K.M.; FOCARDI, S. Technology Overview : A Report on Data Mining, *Technical Report CSCS TR-95-02*, Swiss Scientific Computer Center, 1995.

[DHA00] DHAR, V.; CHOU, D.; PROVOST, F. Discovering interesting patterns for investment decision making with GLOWER - a genetic learner overlaid with entropy reduction. *Data Mining and Knowledge Discovery,* 4. 2000, p.251-280.

[EIJ99] EIJKEL, G.C. Rule Induction, In Berthold, M., Hand, D.J., (Eds.), *Intelligent Data Analysis*, Berkeley, CA: Springer- Verlag. 1999, p.196-216.

[FAL98] FALKENAUER, E. *Genetic Algorithms and Grouping Problems*. West Sussex:John Wiley & Sons Ltd. 1998.

[FAY96] FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P.; UTHURUSAMY, R. *Advances in Knowledge Discovery and Data Mining. American Association for Artificial Intelligence.* Menlo Park, CA: MIT Press. 1996.

[FAY98] FAYYAD, U. Mining Databases: Towards Algorithms for Knowledge Discovery. *Data Engineering IEEE Computer Society*. Washington. 1998, p. 39-48.

[FEE99] FEELDERS, A.J. Statistical Concepts, in Berthold, M., Hand, D.J., (Eds.), *Intelligent Data Analysis*, Berkeley. Springer-Verlag. 1999, p.15-66.

[FIS93] FISHER, D.; HAPANYENGWI, G. Database Management and Analysis Tools onf Machine Induction, *Journal of Intelligence Information Systems, 2,* Kluwer Academic Publihers, Boston. 1993, p. 5-38

[FOG00] FOGEL, D. Other selection methods, in Back, T, Fogel, D.B., Michalewicz (Eds.), *Evolutionary Computation 1,* Philadelphia: Institute of Physics Publishing. 2000, p.201-204.

[FRE98] FREITAS, A.A. LAVINGTON, S.H. *Mining Very Large Databases with Parallel Processing*, MA: Kluwer Academic Publishers. 1998.

[FRE01] FREITAS, A.A. Understanding the Crucial Role of Attribute Interaction in Data Mining. *Artificial Intelligence Review 16(3)*, November, 2001, p.177-199.

[FRI96] FRIEDMAN, J.H.; KOHAVI, R.; YUN,Y. Lazy Decision trees. *Proc. 1996 Nat. Conf. of AAAI* (AAAI-96). 1996, p.717-724.

[FU96] FU, Y. *Discovery of Multiple-Level Rules from Large Databases*, Ph.D. Thesis of Doctor of Philosophy, Faculty of Applied Sciences, Simon Fraser University, British Columbia, Canada. 1996, 184p.

[GIO95] GIORDANA, A.; NERI, F. Search-Intensive Concept Induction, *Evolutionary Computation*, 3(4). 1995, p.337-416.

[GOL87] GOLDBERG, D.E.; RICHARDSON, J. Genetic Algorithms with Sharing for Multimodal Function Optimization. *Proc. Of 2nd Int. Conf. On Genetic Algorithms (ICGA-87).* 1987, p.41-49.

[GOL89] GOLDBERG, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989.

[GOL92] GOLDBERG, D.E.; HORN, J.; DEB, K. What Makes a Problem Hard for a Classifier System? *Illigal Report N. 92007*. May, 1992.

[GOL97] GOLDBERG, D.E.; WANG, L. Adaptative Niching Via Coevolutionary Sharing. *Illigal Report N. 97007*. August, 1997.

[GRE93] GREENE, D.P.; SMITH, S. Competition-Based Induction of Decision Models from Examples. *Machine Learning 13*. 1993, p.229-257.

[HAN97] HAND, D.J. *Construction and Assessment of Classification Rules*, New York: John wiley & Sons. 1997.

[HAN99] HAND, D.J. Introduction, In Berthold, M., Hand, D.J., (Eds.), *Intelligent Data Analysis*, Berkeley, CA: Springer- Verlag. 1999, p.1-14.

[HAS00a] HASSE, M. *Mineração de Dados usando Algoritmos Genéticos*, Dissertação de Mestrado. Departamento de Informática, Curitiba, UFPR. Agosto, 2000, 76 p.

[HAS00b] HASSE, M.E.; POZO, A.R. Using phenotypic sharing in a classifier tool. *Proc. Of the Genetic and Evolutionary Computation Conf. - GECCO 2000*. Las Vegas: Morgan Kaufmann. Julho, 2000, p.392.

[HOL94] HOLSHEIMER, M.; SIEBES, A. Data Mining the Search for Knowledge in Databases, *Technical Report CS-R9406*. CWI. Amsterdã. 1994.

[HOL89] HOLTE, R.C.; ACKER, L.E. ; PORTER, B.W. Concept Learning and the Problem of Small Disjuncts, *Proc. Int. Joint Conf. on Artificial Intelligence. IJCAI – 89*. 1989, p.813-818.

[HOR97] HORN, J. *The nature of niching: Genetic Algorithms and the Evolution of optimal, cooperative populations*. Ph.D. Thesis Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign. 1997, 242p.

[HOR94] HORWOOD, E. *Machine Learning, Neural and Statistical Classification*, D. Michie, D.J.Spiegelhalter, C.C. Taylor (Eds). 1994.

[HUN96] HUNT, J.E.; COOKE, D.E. Learning Using An Artificial Immune System, *International Journal of Network and Computer Applications: special issue on Intelligent Systems.* 1996, p.189-212.

[JAN91] JANIKOW, C. Inductive learning of decision rules from attribute-based examples: A knowledge-intensive genetic algorithm approach. *Technical Report TR91-030*, The University of North Carolina at Chapel Hill, Dept. of Computer Science, Chapel Hill, NC. 1991.

[KLO92] KLOSGEN, W. Patterns for Knowledge Discovery in Databases. *Proc. Of Machine Learning*. UK. 1992, p. 1-9.

[KUB98] KUBAT, M.; BRATKO, I.; MICHALSKI, R.S. A Review of Machine Learning Methods, in Michalski, R.S., Bratko, I. and Kubat, M. (Eds.), *Machine Learning and Data Mining: Methods and Applications*, London: John Wiley & Sons. 1998, p.3-69.

[LIU00a] LIU, J.; KWOK, J.T. An extended genetic rule induction algorithm. *Proc. Conf. On Evolutionary Computation - CEC 2000*. Piscataway, NJ: IEEE Press. 2000, p. 458-463.

[LIU00b] LIU, B.; HU, M.; HSU, W. Multi-level Organization and Summarization of the Discovered Rules, *Proc. 6th ACM SIGKDD Int. Conf. On Knowledge Discovered & Data Mining (KDD– 2000)*, New York: ACM Press. 2000, p.208-217.

[LOP96] LOPES, H.S. *Analogia e Aprendizado Evolucionário: Aplicação em Diagnóstico Clínico.* Tese de Doutorado, Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina. 1996, 159 p.

[LOP00] LOPES, A.A.; JORGE, A. Integrating Rules and Cases in Learning via Case Explanation and Paradigm Shift, *Advances in Artificial Intelligence. (Proc. IBERAMIA – SBIA 2000)*. LNAI 1952, Berlin: Springer-Verlag. 2000, p.33-42.

[MAH93] MAHFOUD, S.W. Simple Analytical Models of Genetic Algorithms for Multimodal Function Optimization, *Illigal Report N. 93001*. 1993.

[MAH95] MAHFOUD, S.W. *Niching Methods for Genetic Algorithms,* Illigal Report N. 95001, Ph.D. ,Ph.D. Thesis in Computer Science, University of Illinois, Urbana , IL, USA. 1995, 251p.

[MIC96] MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd Ed. Berlin: Springer-Verlag. 1996.

[MIC98] MICHALSKI, R.S.; KAUFMAN, K.A. Data Mining and Knowledge Discovery: A Review of Issues and Multistrategy Approach. In: Michalski, R.S., Bratko, I. and Kubat, M. (Eds.), *Machine Learning and Data Mining: Methods and Applications*, London: John Wiley & Sons. 1998, p.71-112.

[MIT97] MITCHELL, T. *Machine Learning.* New York: McGraw-Hill. 1997.

[NAZ99] NAZAR, K.; BRAMER, M.A. Estimating concept difficulty with cross entropy. In: Bramer, M.A. (Ed.) *Knowledge Discovery and Data Mining*, London: The Institution of Electrical Engineers. 1999, p.3-31.

[NOD99] NODA, E.; LOPES, H.S.; FREITAS, A.A. Discovering interesting prediction rules with a genetic algorithm. *Proc. CEC-99*, Piscataway, NJ: IEEE Press. 1999, p.1322-1329.

[NOC98] NOCK, R.; JAPPY, P. On the power of Decision Lists. *Proc. Int. Conf. Machine Learning (ICML – 98)*, San Francisco, CA: Morgan Kaufmann. 1998, p.413-420.

[PAP01] PAPAGELIS, A.; KALLES, D. Breeding Decision Trees Using Evolutionary Techniques. *Proc. 18th Int. Conf. Machine Learning*, San Francisco, CA: Morgan Kaufmann. 2001, p.393-400.

[PAW95] PAWLOWSKY, M.A. Crossover Operators, In: Lance Chambers (Ed.), *Pratical Handbook of Genetic Algorithms Aplications I,* , Boca Raton: CRC Press. 1995, p.101-114.

[PRO96] PROVOST, F.; ARONIS, J.M. Scalling Up Inductive Learning with Massive Parallelism. *Machine Learning 23(1).* 1996, p. 33-46.

[PRO99] PROVOST, F. ; KOLLURI, V.A Survey of Methods for Scaling Up Inductive Algorithms. in Fayyad, U. (Ed.), *Data Mining and Knowledge Discovery 2,* Kluwer Academic Publishers. Netherlands. 1999, p.131-169.

[QUI86] QUINLAN, J.R. Induction of Decicion Trees, *Machine Learning*, 1(1). 1986, 81-106.

[QUI87] QUINLAN, J.R. Simplifying decision trees. *International Journal of Man-Machine Studies*, 12. 1987, p. 221-234.

[QUI91] QUINLAN, J.R. Improved Estimates for the Accuracy of Small Disjuncts, *Machine Learning 6(1).* 1991, p. 93-98.

[QUI93] QUINLAN, J.R. *C4.5 Programs for Machine Learning,* San Diego, CA: Morgan Kaufmann Publishers. 1993.

[REN90] RENDELL, L.; SESHU, R. Learning hard concepts through constructive indiction: framework and rationale. *Computational Intelligence* 7. 1990, p.247-270.

[ROM02a] ROMÃO, W.; FREITAS, A.A.; PACHECO, R.C.S. A Genetic Algorithn for Discovering Interesting Fuzzy Prediction Rules:applications to science & tecnology data. To appear in *Proc. 2002 Genetic and Evolutionary Computation Conf. (GECCO-2002),* New York, July, 2002.

[ROM02b] ROMÃO,W. *Descoberta de Conhecimento Relevante em Banco de Dados sobre Ciência e Tecnologia.* Tese de doutorado. Departamento de Engenharia de Produção, Universidade Federal de Santa Catarina. 2002, 238 p.

[RYA95] RYAN, C. Niche and Species Formation in Genetic Algorithms, In: Lance Chambers (Ed.), *Pratical Handbook of Genetic Algorithms Aplications I,* , Boca Raton: CRC Press. 1995, p.57-74.

[SAR00] SARMA, J.; DE JONG, K. Generation gap methods, in Back, T, Fogel, D.B., Michalewicz (Eds.), *Evolutionary Computation 1,* Philadelphia: Institute of Physics Publishing. 2000, p.205-227.

[SCH93a] SCHAFFER, C. Overfitting Avoidance as Bias. *Machine Learning*, 10(2), Kluwer Academic Publisher.1993, p.153-178.

[SCH93b] SCHAFFER, C. Selecting a Classification Method by Cross-Validation. *Machine Learning, 13(1).* Kluwer Academic Publisher. 1993, p. 135-143.

[SKI95] SRIKANT, R.; AGRAWAL, R.Mining Generalized association rules. *Proc. 21 Int. Conf. Very Large Databases*. 1995, p.407-419.

[SMI93] SMITH, R.E.; FORREST, S.; PERELSON, A.S. Searching for Diverse, Cooperative Populations with Genetic Algorithms, *Evolutionary Computation, 1(2).* 1993, p.127-149.

[SMI00] SMITH, R.E. Learning classifier system. In: Back, T, Fogel, D.B., Michalewicz (Eds.), *Evolutionary Computation 1,* Philadelphia: Institute of Physics Publishing. 2000, p.114-123.

[SYS89] SYSWERDA, G. Uniform crossover in genetic algorithms. *Proc.3rd Int.Conf.on Genetic Algorithms (ICGA 89)*, San Mateo, CA: Morgan Kaufmann Pub. 1989, p.2–9.

[TIN94] TING, K.M. The Problem of Small Disjuncts: its remedy in Decision Trees, *Proc. 10th Canadian Conf. on Artificial Intelligence*, Palo Alto, CA: Morgan Kaufman. 1994, p.91-97.

[VEN93] VENTURINI, G. SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. *Proc. Of The European Conf. on Machine Learning*, Lecture Notes in Computer Science 667 Springer. 1993, p.280–296.

[WEB94] WEBB, G. Recent Progress in Learning Decision Lists by Preparing Inferred Rules. Proc. of Second Singapore International Conf. on Intelligent Suystems. 1994, p. 291-298.

[WEI91] WEISS, S.M.; KULIKOWSKI, C.A. *Computer Systems That Learn.* San Mateo, Morgan Kaufmann Publishers, Inc. 1991.

[WEI95] WEISS, G.M. Learning with Rare Cases and Small Disjuncts, *Proc. 12th Int. Conf. on Machine Learning (ICML-95)*, San Francisco, CA: Morgan Kaufmann. 1995, p.558-565.

[WEI98] WEISS, G.M. The Problem with Noise and Small Disjuncts, *Proc. Int. Conf. Machine Learning (ICML – 98)*, San Francisco, CA: Morgan Kaufmann. 1998, p.574-578.

[WEI00] WEISS, G.M.; Hirsh, H. A Quantitative Study of Small Disjuncts, Proc. of Seventeenth Nat. Conf. on Artificial Intelligence (AAAI – 2000). Austin, Texas, Menlo Park, CA: AAAI Press. 2000, p.665-670.

[WET95] WESTTSCHERECK, D.; AHA, D.W. Weighting features. *Proc. 1st Int. Conf. CBR. (ICCBR-95). LNAI 1010*, 347-358. 1995, 347-358.

[WIT99] WITTEN, I.; FRANK, E. *Data Mining – Pratical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers. San Francisco, CA. 1999.

[WU95] WU, X. *Knowledge Acquisition from Databases*, USA: Ablex Publishing Corporation. 1995.