Universidade Federal da Bahia

Instituto de Matemática

Departamento de Ciência da Computação

Mineração de Dados Utilizando Algoritmos Genéticos

Joilma Souza Santos

Orientadora: Profª. Daniela Barreiro Claro

Salvador – Bahia

2008

Joilma Souza Santos

Mineração de Dados Utilizando Algoritmos Genéticos

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal da Bahia como requisito parcial à obtenção do diploma de Bacharel em Ciência da Computação.

Orientadora: Profª. Daniela Barreiro Claro

Salvador – Bahia
2008

AGRADECIMENTOS

Agradeço a Deus por tudo que tem feito por mim e por nunca ter me deixado sucumbir diante das dificuldades que enfrentei para chegar até aqui.

Agradeço a meu pai pelos sacrifícios que fez por mim e pelo amor que sempre me deu e a quem eu sempre serei grata, e a minha irmã Joelma.

A meu namorado Tadeu que me ajudou nos momentos mais difíceis, e que todo dia me oferece suporte.

A Eric Sobral, pela interface gráfica da ferramenta deste projeto, e por sua amizade.

A meus colegas de trabalho, que para mim já são como parte de minha família e que me ajudaram nesse projeto.

A todos os meus amigos e amigas, que me apoiaram com uma palavra de incentivo, com um sorriso quando estava desanimada. Vocês são tesouros em minha vida, e fica aqui o meu mais sincero agradecimento.

Joilma Souza Santos

“Estamos nos afogando em dados,
mas sedentos por conhecimento”.
John Naisbitt

RESUMO

A produção crescente de dados por organizações privadas e públicas produz enormes bases de dados. Conseqüentemente, as tarefas de análise e extração de informações tornam-se extremamente custosas para um analista humano. No processo de Mineração de Dados ou *Data Mining*, a análise e extração do conhecimento é realizada procurando-se padrões consistentes e/ou relacionamentos sistemáticos entre as instâncias destes dados. Para implementação dessa técnica, métodos automáticos baseados na Inteligência Artificial são usados a fim de aperfeiçoar o processo de análise. Este projeto visa determinar padrões através dos algoritmos genéticos com o intuito de minerar os dados. Os padrões obtidos através dos algoritmos genéticos são comparados com padrões determinados através do uso de um outra técnica de Inteligência Artificial, a Lógica *Fuzzy* e uma ferramenta com o intuito de integrar estas duas técnicas foi desenvolvida.

Palavras-Chave: Mineração de Dados, KDD, Inteligência Artificial, Algoritmos Genéticos, Banco de Dados.

ABSTRACT

The growing volume of information by public and private organizations produces huge databases. Therefore, the tasks of analysis and extraction of information becomes extremely costly for analysts. In the process of Data Mining, these tasks require consistent standards and / or systematic relationships between different instances of these data. To implement and optimize the analysis process, automation methods based on artificial intelligence are used. This project aims to determine patterns through genetic algorithms with a view to mine the data. The patterns obtained from the Genetic algorithms are compared with standards set by use a different technique of artificial intelligence, fuzzy logic and a tool with the aim of integrating these two techniques was developed.

Keywords: Data Mining, KDD, Artificial Intelligence, Genetics Algorithms, Data Bases

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

KDD	*Knowledge Data Discovery*	p.13
WEKA	*Waikato Environment for Knowledge Analysis*	p.13
EFT	*Explorer Fuzzy Tree*	p.13
AG	Algoritmo Genético	p.24
COGIN	Covered-based Genetic Induction	p.39
REGAL	Relational Genetic Algorithm Learner	p.40
VP	Verdadeiro Positivo	p.44
VN	Verdadeiro Negativo	p.44
FP	Falso Positivo	p.44
FN	Falso Negativo	p.44
EPT	*Explorer Patterns Tool*	p.49

SUMÁRIO

1 INTRODUÇÃO

O advento do desenvolvimento da tecnologia tornou a captação e armazenamento de dados uma tarefa mais simples e barata (FIGUEIRA, 2008 apud FERNANDES, 2003a), o que contribuiu para o aparecimento de bases de dados cada vez mais complexas e maiores.

Com a crescente valorização da informação nas últimas décadas, a extração de conhecimento desses dados brutos torna-se um diferencial no desenvolvimento de estratégias de investimento de empresas públicas ou privadas. Experiências como a relatada no lendário exemplo das fraldas e cerveja mostram como decisões baseadas na análise de dados podem aumentar o volume de vendas de determinado produto (FUCHS, 2004).

O modo mais tradicional para extrair informação de uma base de dados baseia-se na análise e interpretação manuais; este modo de análise, além de lento, é computacionalmente custoso e subjetivo (FAYYAD et al., 1996). Visando a extração mais rápida e confiável de conhecimento, surgem métodos automáticos denominados *Knowledge Data Discovery*, ou simplesmente KDD. O KDD aplica métodos interdisciplinares – especialmente métodos estatísticos e de aprendizado de máquina para extrair conhecimento de alto nível a partir de bases de dados reais (FREITAS, 2001) e tem a Mineração de Dados como principal tarefa.

O processo de Mineração de Dados ou *Data Mining* analisa e extrai informações úteis a partir de dados. A captação de conhecimento de alto nível através deste processo se dá a partir da procura de padrões consistentes e/ou relacionamentos sistemáticos entre instâncias de dados, classificando-os em subconjuntos de informações baseados em regras estatísticas e da teoria da informação.

Algoritmos de aprendizado de máquina são amplamente utilizados na tarefa de Mineração de Dados. Estes algoritmos são baseados na construção de árvores de decisão, utilizando-se como matéria prima os dados de treinamento. Através do percurso destas

árvores, da raiz até um nó folha[1] é possível inferir classes a dados baseados em determinados valores para cada atributo.

Para uma classificação com maior percentual de acerto (acurácia[2]), métodos de aprendizado de máquina baseados em *Lógica Fuzzy* têm-se mostrado mais eficientes que métodos clássicos (SOUZA, 2008). Apesar da perceptível contribuição que a *Lógica Fuzzy* proporciona à acurácia das árvores de classificação utilizadas na Mineração de Dados, o WEKA (*Waikato Environment for Knowledge Analysis*) (FRANK; TRIGG, 1993), principal ferramenta gratuita para Mineração de Dados, baseada em algoritmos de aprendizado de máquina, não implementa técnicas com algoritmos baseados na lógica nebulosa.

Outra abordagem para realização da Mineração de Dados utiliza algoritmos de aprendizado de máquina baseados na Computação Evolutiva – os Algoritmos Genéticos. Estes algoritmos realizam buscas globais e permitem uma melhor interação entre atributos que os algoritmos baseados na estratégia gulosa que são geralmente utilizados para a tarefa de Mineração de Dados e por isso são extremamente úteis na procura de padrões relevantes em dados (FREITAS, 2001 apud DHAR V; PROVOST, 2000). Os Algoritmos Genéticos também não foram implementados no WEKA.

Para disponibilizar a Mineração de Dados utilizando árvores de classificação *fuzzy* e testar a acurácia dessas árvores em relação a árvores de classificação clássicas, foi desenvolvido o EFT (*Explorer Fuzzy Tree*) (SOUZA, 2008). Para determinar a acurácia das árvores *fuzzy* implementadas pelo EFT, foram feitos testes utilizando quatro bases de dados de complexidades diferentes: Iris (FISHER, 1988), Segment-challenge (GROUP, 1990a), Segment-test (GROUP, 1990b) e SPAMBASE (HOPKINS et al., 2001).

Assim, o presente trabalho aplica Algoritmos Genéticos à tarefa de Mineração de Dados e incorpora o algoritmo implementado à ferramenta *Explorer Fuzzy Tree* (EFT). Com o intuito de avaliar o algoritmo implementado, novos experimentos foram realizados sobre estes algoritmos utilizando as mesmas quatro bases de dados utilizadas para o teste dos

[1] Nó folha – nó terminal de uma árvore, que no caso de uma árvore de classificação, corresponde a classe que caracteriza um dado objeto com os valores de atributos;
[2] Acurácia – exatidão de uma operação;

algoritmos de construção de árvores *fuzzy*. Os resultados obtidos mostraram uma igualdade de desempenho entre as duas técnicas para bases de dados médias,

Esta monografia está dividida em seis capítulos, dispostos da seguinte forma: o capítulo 2 apresenta e detalha os conceitos de Mineração de Dados; o capítulo 3 apresenta os Algoritmos Genéticos e a Computação Evolutiva; o capítulo 4 detalha a implementação do algoritmo genético usado para extração de conhecimento de alto nível de bases de dados; o capítulo 5 apresenta os experimentos realizados e discute seus resultados; e por fim, capítulo 6 apresenta a conclusão e trabalhos futuros.

2 MINERAÇÃO DE DADOS

Conhecimento é a informação devidamente organizada e que pode ser aplicada a resolução de problemas. Em uma base de dados, encontra-se a matéria-prima para a obtenção de conhecimento, que pode ser realizada utilizando o processo de descoberta de conhecimento em bases de dados, ou KDD.

Atualmente, as bases de dados vêm crescendo em alta velocidade, abrigando milhões de registros com dezenas de atributos. Essas bases de dados crescem de suas maneiras: o número N de registros ou objetos na base aumenta ou ainda o número d de campos ou atributos deste objeto aumenta, de modo que bases de dados que possuem em média N = 10^9 objetos ou d = 10^2 ou até mesmo d = 10^3 têm-se tornado comum (FAYYAD et al., 1996). A análise manual dessas bases de dados com grandes números de atributos ou registros pode ser um processo demorado.

Os resultados da análise destes dados armazenados podem ser utilizados para atribuir maior competitividade às empresas, que, baseadas nestes resultados, podem desenvolver estratégias mais efetivas para obtenção de sucesso em suas respectivas áreas de atuação, assim como também pode ser utilizada no campo científico, para promover a melhor interpretação dos dados coletados através das pesquisas científicas.

O KDD é um processo não trivial de identificação de padrões válidos, potencialmente úteis e compreensíveis a partir de dados[3] (FAYYAD et al., 1996). Para a identificação destes padrões, o processo de KDD realiza operações em dados brutos que vão desde a seleção dos dados a serem utilizados até a análise dos padrões encontrados para a obtenção de conhecimento útil. A busca destes padrões possui algumas etapas baseadas em estratégicas

[3] Segundo Fayyad, dado é um conjunto de fatos e padrão é uma expressão em alguma linguagem que descreve um subconjunto de dados ou um modelo aplicável a este subconjunto; o termo processo denota que o KDD envolve vários passos; não trivial denota que o processo de KDD demanda alguma busca ou inferência.

computacionais, o que permite que enormes volumes de dados sejam analisados rapidamente e de forma não subjetiva. A figura 2.1 apresenta as etapas do processo de KDD.

Figura 2.1: Fases do Processo de KDD (PAPPA, 2002 apud LIU; MOTODA, 1998)

Há ainda confusão em relação aos termos Mineração de Dados e KDD. Esses termos são esporadicamente tratados como sinônimos. Porém, como mostrado na figura acima, a Mineração de Dados é apenas uma das etapas do processo de KDD.

A Mineração de Dados consiste na aplicação de algoritmos para a análise e descoberta de dados e na produção de padrões ou modelos a partir de grandes bases de informação (FAYYAD et al., 1996), enquanto que o termo KDD refere-se a todos os passos aplicados a dados brutos para a obtenção de dados de alto nível (GOEBEL; GRUENWALD, 1999). As fases do processo de KDD são discutidas a seguir.

2.1 As Fases do Processo de Descoberta de Conhecimento

O processo de KDD é interativo e iterativo e envolve vários passos onde decisões são feitas pelo usuário (FAYYAD et al., 1996). A primeira etapa do processo de KDD é a etapa de *Data Warehouse*, que coleta e reuni informações de várias bases de dados, realizando a integração entre eles. No pré-processamento, as informações coletadas na fase de *Data Warehouse* são filtradas e é selecionado apenas o conteúdo relevante para a tarefa que será realizada. Este processo de filtragem é reforçado por métodos de redução de ruídos, correção de erros e preenchimento de valores nulos, para garantir a confiabilidade dos dados a serem utilizados.

É geralmente desejável a intervenção de um analista humano nas etapas citadas anteriormente, a fim de que seu conhecimento sobre as informações relevantes que devem ser extraídas pelo processo de KDD seja utilizado (FREITAS, 2001).

Após o refinamento dos dados a serem utilizados, a discretização pode ser utilizada para transformar um atributo com valores contínuos em um atributo categórico (*categorical*) ou nominal, formando intervalos discretos (FREITAS, 2001).

A etapa de seleção de atributos encontra atributos que possuem relevância para o objetivo que deve ser satisfeito pelo processo de KDD, encontrando um subconjunto de atributos relevantes (atributos capazes de distinguir exemplos pertencentes a classes diferentes (PAPPA, 2002)) entre os atributos originais do registro para ser utilizado no processo de mineração de dados. A motivação para a realização desta etapa é baseada nos seguintes fatores: algoritmos de mineração de dados que utilizam métodos de indução tem seu tempo de execução aumentado caso haja muitos atributos presentes (PAPPA, 2002); a presença de atributos irrelevantes ou redundantes pode de alguma maneira levar o algoritmo a uma classificação equivocada dos dados, levando o processo a produzir dados pouco relevantes ou não confiáveis (FREITAS, 2001); experimentos comprovam que o número de exemplos para garantir certa taxa de classificação cresce de maneira exponencial baseada no número de atributos irrelevantes presentes (PAPPA, 2002 apud LANGLEY; IBA, 1993). Outra motivação encontrada para a seleção de atributos está na melhora da performance do algoritmo de mineração de dados, com melhores taxas de aprendizado ou regras mais simples (PAPPA, 2002).

A próxima etapa do processo de KDD é a Mineração de Dados. Nesta etapa é aplicado um algoritmo baseado na estratégia computacional escolhida para mineração de dados, e é extraído conhecimento dessa base de dados. A última etapa do processo, o pós-processamento, visa validar, interpretar e organizar o conhecimento encontrado, obtendo conhecimento realmente útil (PAPPA, 2002).

É interessante salientar que o conhecimento resultante do processo de KDD deve possuir as seguintes propriedades: ser confiável (conhecimento com alta acurácia), compreensível e interessante (FREITAS, 2001). Conhecimento com alta taxa de acurácia nos

permite seu uso para prever o valor que alguns atributos poderão possuir no futuro, baseado nos dados observados. Este conhecimento deve também ser compreensível para o usuário, já que ele deve utilizar o conhecimento adquirido no processo como diferencial em várias áreas. Além disso, esses dados devem ser interessantes – propriedade mais difícil de ser medida pelo seu caráter subjetivo

2.2 Tarefas da Mineração de Dados

Essa sessão discute algumas técnicas utilizadas pela Mineração de Dados: regras de classificação, a clusterização e as regras de associação.

2.1.1 Regras de Classificação

Alvo de estudo das comunidades da área estatística e de aprendizado de máquina, a classificação tem como objetivo prever o valor de um atributo objetivo, a ser determinado por uma referência externa, baseado no valor dos demais atributos de um exemplo *E* pertencente a uma base de dados *B*. A figura 2.2 demonstra o processo de classificação aplicado a uma base de dados.

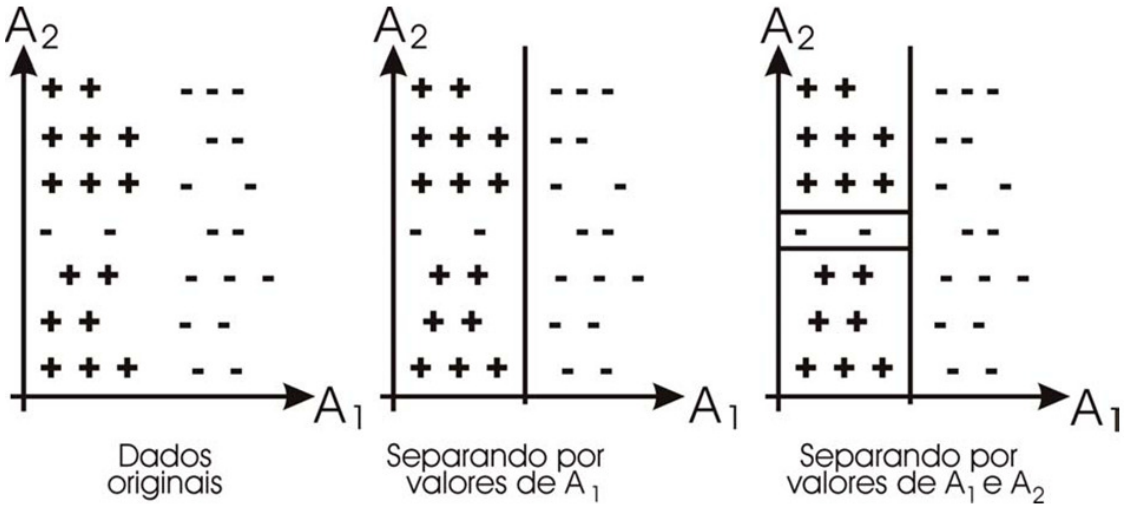

Figura 2.2: Exemplo de Classificação (CARVALHO, 2005)

Um algoritmo de classificação gera regras do tipo $X \rightarrow Y$ (lê-se: Se *X* então *Y*), onde *Y* (a parte conseqüente da regra) representa o atributo objetivo (classe) e *X* (a parte

antecedente da regra) representa um conjunto de valores tomados por atributos, geralmente representados por uma conjunção. A seguinte regra poderia ser uma regra gerada por um algoritmo de classificação: $(idade > 35)\ E\ (salario > R\$\ 2.500{,}00) \rightarrow risco_{emprestimo} = baixo$, onde $(idade > 35)\ E\ (salario > R\$\ 2.500{,}00)$ seria a parte antecedente da regra e $risco_{emprestimo} = baixo$ seria a parte conseqüente da regra (classe).

Este trabalho utiliza algoritmos genéticos para gerar regras que terão o formato descrito acima. Este formato foi escolhido por ser intuitivo e facilmente compreendido pelo usuário. Abaixo outros exemplos de regras de classificação:

$$(tempo = ensolarado)\ E\ (25° < temperatura < 30°) \rightarrow jogar = sim.$$

$$(remetente = não\ informado\ )\ E\ (tipoArquivoAnexo = executável) \rightarrow span = sim$$

Para a geração das regras de classificação, a base de dados é dividida em dois conjuntos de exemplos: C_1 e C_2, que possuem todos os seus exemplos já previamente classificados. O algoritmo de classificação recebe o conjunto de exemplos C_1 (o conjunto de treinamento), aplica técnicas de estatísticas e/ou de aprendizado de máquina e gera as regras de classificação, baseado nos valores encontrados nos atributos de cada um dos exemplos. Posteriormente, o algoritmo aplica as regras de classificação geradas no conjunto de exemplos C_2 (o conjunto de teste), e mede o quão confiáveis são as regras geradas no processo anterior. A propriedade que mede o quão confiável uma regra é, ou seja, quantos exemplos do conjunto de teste foram corretamente classificados por esta regra, é a acurácia. É importante que o algoritmo de classificação não tenha acesso ao conjunto de teste na etapa de geração de regras, ou então, a medição da acurácia das regras ficaria comprometida e a fase de testes não apresentaria resultados confiáveis.

2.1.2 Clusterização (Agrupamento)

A clusterização visa classificar a informação a ser minerada em clusters ou classes. Cada um desses clusters é formado por exemplos que possuem características similares em seus atributos. Essa classificação é feita de modo a maximizar as diferenças encontradas

entre clusters diferentes e minimizar as diferenças entre exemplos pertencentes ao mesmo cluster.

Por realizar a classificação dos exemplos baseado nos valores dos atributos dos próprios exemplos, ou seja, “descobrir” classes para os exemplos e agrupá-los em classes sem que seja informada nenhuma referência externa, a clusterização é considerada uma forma de aprendizado não supervisionado (FREITAS, 2001).

A clusterização prepara os dados para a aplicação de um algoritmo de classificação, uma vez que divide estes dados em classes, que serão usadas posteriormente pelo algoritmo de classificação (CARVALHO, 2005).

Por exemplo, numa base como a Iris (FISHER, 1988), cada exemplo possui determinado valor para os comprimentos e larguras da pétala e da sépala de uma determinada flor. Baseado nesses atributos, um algoritmo de clusterização encontraria três clusters: o cluster Iris-setosa, o cluster Iris-versicolor e o cluster Iris-virginica.

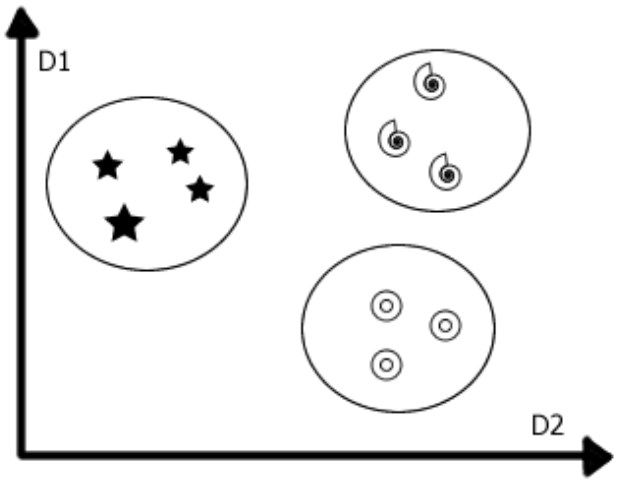

Figura 2.3: Exemplo de dados organizados em clusters (CARVALHO, 2005)

2.1.3 Regras de Associação

As regras de associação são regras encontradas a partir de um conjunto de exemplos *E* do tipo $X \rightarrow Y$ (Se *X* então *Y*), onde *X* e *Y* são conjuntos de itens, tal que $X \cap Y = \emptyset$. Cada

regra encontrada sinaliza que os conjuntos de itens *X* e *Y* são freqüentemente encontrados em um mesmo exemplo.

O uso de regras de associação pode ser ilustrado usando-se o exemplo das fraldas e das cervejas (FUCHS, 2004). Estas regras aplicadas a um conjunto de exemplos *E*, onde cada exemplo representa uma transação de compra de um cliente, permitiram a descoberta do produto cerveja como sendo o mais vendido junto ao produto fraldas, ou seja, foram encontrados muitos exemplos onde os itens fraldas e cervejas faziam parte da compra dos consumidores do supermercado e assim pôde ser comprovada a associação entre a compra das fraldas e a compra das cervejas, ou seja: $Fralda \rightarrow Cerveja$.

Apesar de possuírem a mesma estrutura $X \rightarrow Y$, as regras de associação e regras de classificação possuem diferenças decisivas: regras de associação podem possuir mais de um item na parte conseqüente da regra (*Y*), enquanto regras de classificação só podem possuir um atributo objetivo. Além disso, em regras de classificação, os atributos previsores (*X*) só podem ocorrer na parte antecedente da regra e o atributo objetivo (*Y*) somente ocorre na parte conseqüente da regra, o que não é uma constante em regras de associação. Abaixo alguns exemplos de regras de associação:

- $Pão \rightarrow Manteiga, Queijo$ (clientes que compram pão, geralmente compram manteiga e queijo)

- $Batatas\ Fritas \rightarrow Refrigerantes$ (clientes que compram batatas fritas geralmente compram refrigerante).

2.3 Utilização de Algoritmos Genéticos (AGs) no Processo de KDD

Os algoritmos genéticos podem ser aplicados às etapas do processo de KDD, desde o pré-processamento dos dados brutos para aplicação da mineração de dados – por exemplo, na seleção de atributos, podendo ser aplicados também à mineração de dados. Esta técnica pode também ser aplicada a etapa de pós-processamento do conhecimento encontrado.

O uso dos AGs na etapa de seleção de atributos é motivado pela interação de atributos. Os algoritmos baseados em outras técnicas desenvolvidos para esta tarefa, como

por exemplo, algoritmos baseados na estratégia gulosa, podem não ser confiáveis, levando a não seleção de atributos relevantes para a correta classificação dos dados, como demonstra o exemplo a seguir, ilustrado na figura 2.4.

A_1	A_2	B
0	0	0
0	1	1
1	0	1
1	1	0

Figura 2.4: Exemplo de interação de atributos em problema de classificação do tipo XOR (PAPPA, 2002)

Na figura 2.4 é ilustrado um exemplo utilizando a função XOR. Esta função classifica os dados em 0 ou 1, baseado no valores de todos os seus atributos previsores. Nesta função, dados com valores de atributos previsores (A_1 e A_2) iguais pertencem a classe 0 (B = 0) e dados com valores de atributos diferentes pertencem a classe 1 (B = 1). Caso um método de seleção de atributos guloso fosse aplicado a este exemplo, encontraria uma distribuição igual de classes ao selecionar qualquer um dos atributos previsores, tendo 50% dos resultados igual a 1 e os outros 50% dos resultados iguais a 0 e chegaria a conclusão que nenhum dos atributos previsores são relevantes para a tarefa de classificação, chegando então a uma conclusão equivocada, já que o resultado da função XOR depende de todos os atributos previsores (PAPPA, 2002).

A principal motivação para o uso de algoritmos genéticos para descoberta de regras de classificação também está na melhor interação entre atributos proporcionada pelos AGs se comparados com algoritmos baseados na estratégica gulosa para indução de regras de classificação, e que são geralmente mais utilizados para mineração de dados (FREITAS, 2001). Outra vantagem importante do uso de AGs para esta tarefa é a busca global realizada por esta técnica, aumentando a probabilidade de se obter um conjunto de regras com alta acurácia preditiva.

Duas características dos AGs são decisivas para a sua utilização no processo de KDD, e seu sucesso diante de algumas das técnicas tradicionais de busca: a) algoritmos genéticos

são métodos de busca globais – AGs não utilizam apenas informação local e por isso não necessariamente prendem-se a soluções ótimas locais, como certos métodos de busca (LINDEN, 2006); b) AGs utilizam informação da população corrente para determinar o próximo estado da busca (LINDEN, 2006), e através de uma função objetivo e dos operadores genéticos de cruzamento (*crossover*) e mutação (*mutation*) promovem a interação entre os atributos de um objeto.

Vale salientar que os algoritmos genéticos não garantem que a solução encontrada seja a solução ótima para o problema proposto, todavia estes algoritmos tendem a encontrar boas soluções ou soluções muito próximas da solução ótima. Os operadores genéticos e demais aspectos dos AGs são descritos no capítulo 3 deste trabalho.

3 ALGORITMOS GENÉTICOS (AG)

Algoritmos Evolucionários são basicamente algoritmos inspirados nos princípios da seleção natural e da genética natural (FREITAS, 2007). Existe uma grande variedade de modelos computacionais propostos dentro deste paradigma, mas todos simulam os mecanismos de evolução natural das espécies, onde, a partir de operadores genéticos – seleção, cruzamento e mutação – novas espécies são criadas novas gerações utilizando indivíduos que são avaliados segundo seu desempenho dentro de um ambiente (LINDEN, 2006).

Durante os anos de 1950 e 1960 alguns cientistas computacionais estudaram as técnicas evolucionárias, para que se tornassem uma alternativa de ferramenta de otimização para a resolução de problemas de engenharia, criando um conjunto de soluções candidatas a resolução do problema proposto, e utilizando operadores inspirados na genética e solução natural para a busca das melhores soluções (MITCHELL, 1996). Os operadores genéticos são aproximações computacionais de fenômenos naturais como a reprodução sexuada (crossover ou cruzamento) e a mutação genética (mutação ou mutation) (LINDEN, 2006)

O conceito da evolução natural é aplicado à computação para a resolução de problemas computacionais, pois os mecanismos de evolução parecem se adequar a estes problemas nas mais variadas áreas (MITCHELL, 1996). Problemas que envolvem buscas em um espaço muito grande de solução – como a busca de um conjunto de regras de classificação a partir de uma base de dados, dentre outros problemas que requerem soluções difíceis de serem projetadas, podem utilizar-se do conceito da seleção e evolução natural para criarem e melhorarem soluções adaptadas a esses problemas complexos.

O comportamento padrão dos algoritmos evolucionários é representado pela listagem 3.1.

Listagem 3.1: Pseudocódigo de um Algoritmo Evolucionário (LINDEN, 2006)

```
T:=0                                          //Inicializamos o contador de tempo
Inicializa_População P(0)                     //Inicializamos a população aleatoriamente
Enquanto não terminar faça                    //Condição de término: por tempo, por avaliação, etc.
        Avalie_População P(t)                 //Avalie a população neste instante
        P':= Selecione_Pais P(t)              //Selecionamos sub-população que gerará nova geração
        P' = Recombinação_e_mutação P'        //Aplicamos os operadores genéticos
        Avalie_População P'                   //Avalie esta nova população
        P(t+1)=Selecione_sobreviventes P(t), P'   //Selecione sobreviventes desta geração
        t:= t+1                               //Incrementamos o contador de tempo
Fim enquanto
```

Como ilustrado na listagem 3.1, os algoritmos evolucionários são métodos fortemente probabilísticos, visto que a geração da população inicial e a seleção feita para a escolha dos indivíduos que serão recombinados e submetidos à mutação raramente gera indivíduos iguais a cada execução do algoritmo, o que faz com que os resultados encontrados por um algoritmo evolucionários sejam raramente reprodutíveis, e por isso são heurísticas[4] que não asseguram a obtenção do melhor resultado possível em todas as suas execuções (LINDEN, 2006)

Em 1975, John Holland no livro "*Adaptive in Natural and Artificial Systems*", formalizou e fundamentou matematicamente os algoritmos genéticos. Mesmo não tendo sido o primeiro a aplicar os conceitos da evolução natural à programação, John Holland foi o primeiro a provar matematicamente a eficácia da estratégia evolucionária em problemas de busca. Em seu trabalho, Holland apresenta os algoritmos genéticos como uma abstração dos processos evolutivos, que permitiriam importar os conceitos de adaptação, evolução e seleção natural da vida real para o mundo computacional, a fim de resolver problemas que envolvem a busca por uma solução ótima (LINDEN, 2006).

[4] Heurísticas são algoritmos polinomiais que não podem garantir que a solução encontrada para o problema proposto é a melhor solução, mas que usualmente tentem a encontrar a solução ótima ou próxima da ótima (Linden, 2006).

Os algoritmos genéticos são uma técnica heurística de otimização global baseada no processo biológico da evolução natural (LINDEN, 2006). O grande diferencial desta técnica é a sua capacidade de não se restringir a máximos locais, como outros métodos de otimização, explorando o espaço de busca como um todo.

A maioria dos algoritmos que tratam problemas de otimização não são capazes de encontrar uma solução ótima global, e se restringem a ótimos locais, como o *hill climbing*, que seguem a derivada de uma função e facilmente se prende a máximos locais, desprezando o máximo global, como mostrado na figura abaixo. Note que a figura ilustra que numa técnica de *hill climbing* se inicia em qualquer um dos pontos de início marcados, seguirá a direção de maior crescimento e acabará presa no ponto de máximo local, onde a derivada é zero.

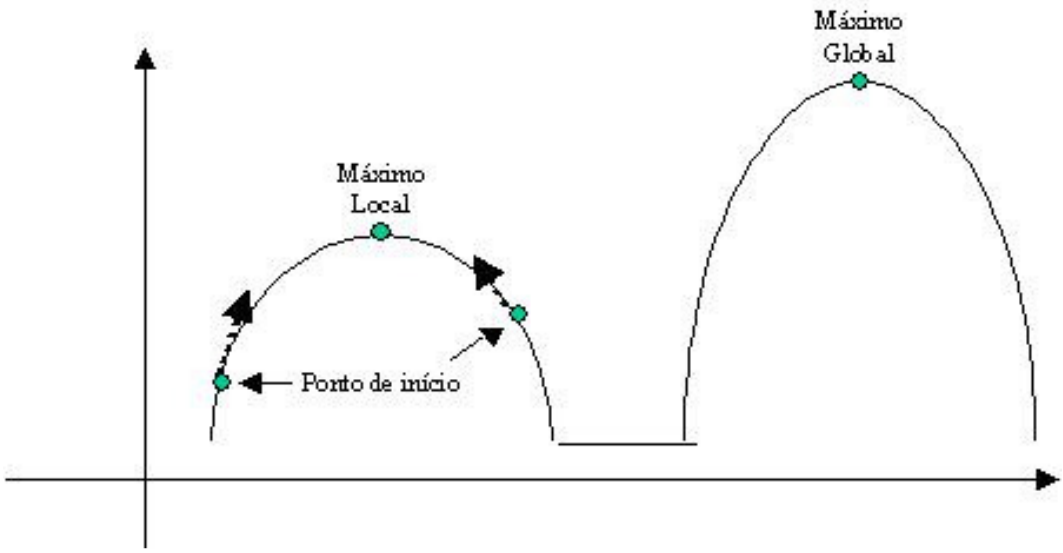

Figura 3.1: Função hipotética com um máximo local e outro global (LINDEN, 2006).

Um algoritmo genético é fundamentado na técnica de geração e teste. Nesta técnica, representada pela figura 3.2, uma solução é gerada; é testada sua eficácia na resolução do problema proposto, considerando limitações impostas, e se por acaso esta solução for adequada à resolução deste problema e obedece às limitações previamente determinadas, ela é adotada. Se por acaso ela não se adéqüe as limitações ou não solucione de maneira satisfatória o problema proposto, ela é desprezada e o processo recomeça gerando uma nova solução a ser testada.

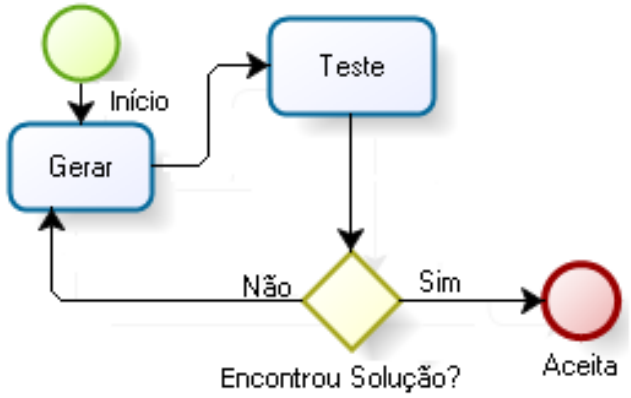

Figura 3.2: Modelo da Técnica de Geração e Teste

A abordagem supracitada utiliza sempre um critério na avaliação destas soluções, e por isso é chamada de busca dirigida. Um algoritmo genético é a utilização da mecânica da genética e da seleção natural à busca dirigida, encontrando os melhores conjuntos de parâmetros que descrevem uma função de adaptação (*fitness*) (FREITAS, 2001).

A fundamentação para a utilização de algoritmos genéticos a problemas de otimização está nos conceitos da seleção natural e evolução das espécies. Segundo esses conceitos, os indivíduos mais adaptados ao seu ambiente vivem tempo o suficiente para se reproduzirem, enquanto os indivíduos menos adaptados morrem antes da reprodução.

Operadores genéticos da seleção natural como cruzamento ou recombinação (*crossover*), mutação (*mutation*) e o uso de uma função de adaptação (*fitness*) para gerar sucessivas gerações de soluções, são aplicados para se chegar à solução que, se não a ótima, é a solução próxima da ótima. Este processo de geração de populações através de algoritmos genéticos para chegar à solução mais adequada na resolução do problema proposto, utilizando o *fitness* de um cromossomo, é chamado convergência (COX, 2005). Cada um destes operadores genéticos é descrito mais profundamente na subseção seguinte.

3.1 Funcionamento de um Algoritmo Genético

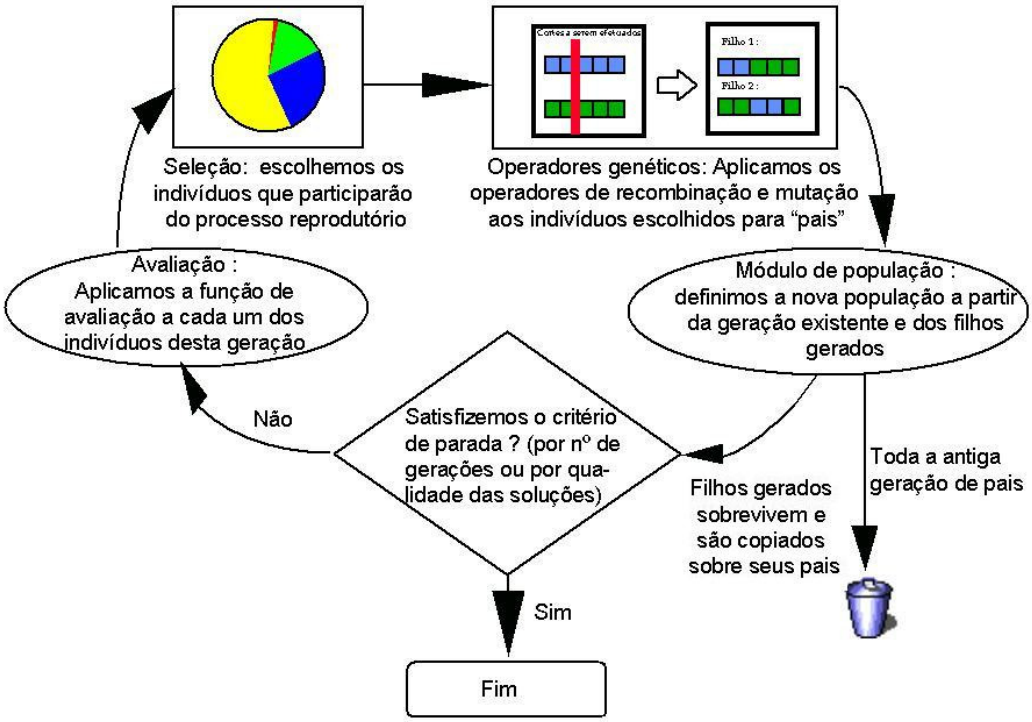

Figura 3.3: Esquema de um algoritmo genético (LINDEN, 2006)

Como ilustrado no esquema 3.3, um algoritmo genético gera uma população inicial *P(N)* com *N* indivíduos (cromossomos), que representam possíveis soluções para o problema a ser atacado. Cada um desses indivíduos *P(n)* é avaliado por uma função que recebe o nome de função de avaliação ou *fitness*. Quanto mais próximo da solução ótima, ou seja, o quanto mais adaptado for o indivíduo, melhor é a avaliação calculada por essa função para este indivíduo. Após a análise de cada um dos indivíduos, a população é analisada e verifica-se se a condição de parada foi satisfeita. Caso a condição de parada não tenha sido satisfeita, *K* indivíduos da população são escolhidos para o processo de reprodução, onde são aplicados os operadores genéticos de recombinação (*crossover*), que gera a partir da combinação de dois indivíduos, como uma analogia a reprodução sexuada, novos indivíduos e/ou a mutação desses indivíduos, e os *N-K* indivíduos são descartados. Após a geração da nova população *P(N)* o processo é refeito. Nas seções seguintes são descritos os aspectos de um algoritmo genético.

3.2 Representação Cromossomial (Codificação do Indivíduo)

Em um algoritmo genético, cada solução candidata é representada por um indivíduo (cromossomo), que é um ponto do espaço de busca dentre todas as soluções possíveis de um problema (CARVALHO, 2005).

Um cromossomo é uma coleção de genomas[5] formado por vários genes (analogia com um cromossomo biológico). Cada gene representa um pedaço indivisível da representação cromossomial (LINDEN, 2006).

Um cromossomo representa uma solução potencial para o problema a ser abordado (COX, 2005). A representação cromossomial deve representar adequadamente o problema em questão, traduzindo suas informações para pra uma maneira viável a ser tratada por um computador. A adaptação destas informações está fortemente ligada à qualidade das soluções obtidas (LINDEN, 2006).

Existem várias maneiras de representação de cromossomos em um algoritmo genético. A escolha de uma representação é arbitrária, e deve ser feita a fim de ser a mais adequada possível a solução do problema, e não o contrário.

A representação mais comumente usada, que também é a representação proposta por Holland é a representação binária de tamanho fixo, onde cada indivíduo é formado por uma cadeia de *bits* que podem assumir os valores 0 ou 1 (PAPPA, 2002 apud Hinterding 2000). Esta representação possui algumas vantagens: além de ser uma representação compacta, facilita os operadores genéticos *crossover* e mutação (COX, 2005). Por outro lado possui alguns problemas, dentre os quais a dificuldade encontrada para a representação de valores contínuos é a principal. Para representar valores contínuos utilizando a representação binária, é necessária a introdução de transformações ou discretizações no algoritmo genético, representando um maior custo a sua execução.

Para a Mineração de Dados a representação cromossomial é geralmente uma seqüência linear de condições de regras, onde usualmente cada condição é um par atributo-valor (CARVALHO, 2005).

[5] Um genoma representa uma característica particular em um cromossomo, e sua natureza depende da representação cromossomial escolhida para o algoritmo genético.

3.3 Função de Avaliação (*Fitness*)

A função de avaliação é a maneira utilizada pelos algoritmos genéticos para determinar a qualidade de um indivíduo como solução do problema em questão (LINDEN, 2006). A função de avaliação e a representação cromossomial estão diretamente ligadas ao problema a ser abordado – caso esses elementos não sejam boas representações deste problema a solução encontrada pode não ser a solução esperada.

Um AG é uma busca dirigida controlada pela função de avaliação (COX, 2005). Cada indivíduo da população de soluções é avaliado segundo esta função e recebe uma nota. Esta nota é utilizada para a escolha dos indivíduos que se reproduzirão, através de um método de seleção que favorece a escolha de indivíduos melhor avaliados (LINDEN, 2006). O *fitness* de um cromossomo depende do quão aquele cromossomo está apto para solucionar o problema em questão (MITCHELL, 1996).

3.4 Esquemas

Introduzidos por Holland, os esquemas formalizam a existência de "blocos construtivos" ("*building blocks"*) em um AG e justificam o seu funcionamento. Esquemas podem ser descritos como um modelo de 0 e 1 e asteriscos, onde os asteriscos representam as posições deste modelo onde o valor ali representado não possui relevância para a solução (*don't cares*) (MITCHELL, 1996). O esquema H = 1 * * * * 1, por exemplo, representa um modelo de todos os cromossomos que se iniciam e terminam com 1, e possuem tamanho de 6 bits. Indivíduos como **1** 0 1 0 0 **1** ou **1** 0 1 1 1 **1** são chamados de instâncias do esquema H.

Cada esquema representa 2^m cromossomos, onde m é o número de posições sem relevância, os "*don't care*" deste esquema. Vale salientar que o valore de 2^m cromossomos representados por um esquema de m posições "*don't care*" é válido apenas para um esquema baseado na representação binária. Para uma população que possui indivíduos de tamanho k, possui-se apenas 3^k esquemas possíveis.

Em um AG, um esquema pode ser avaliado utilizando-se a média dos *fitness* dos indivíduos pertencentes ao esquema em questão. O operador de seleção tende a favorecer

esquemas que possuem avaliações acima da média, de maneira que a quantidade desses esquemas cresça de forma exponencial (CASTRO; ZUBEN, 2002 apud MICHALEWICZ, 1996). Esta afirmação é provada pela equação do crescimento reprodutivo do esquema, representada na figura 3.4.

$$\xi(S_i, t+1) \geq \frac{\xi(S_i, t)\text{eval}(S_i, t)}{\overline{F}(t)}\left[1 - p_c \frac{\delta(S_i)}{m-1} - o(S_i)p_m\right]$$

Figura 3.4: Equação do crescimento reprodutivo do esquema (CASTRO; ZUBEN, 2002).

Seja o esquema S_i. $\xi(S_i, t)$ representa o número de cromossomos representados pelo esquema S_i na geração *t*. $\overline{F}(t)$ representa o *fitness* médio da população na geração *t*. As variáveis p_c e p_m representam as probabilidades de ocorrência de cruzamento e mutação para o AG em questão, respectivamente e m o tamanho de cada indivíduo (tamanho da cadeia de genes). Nesta equação, considera-se que o AG produz indivíduos que possuem avaliações positivas (CASTRO; ZUBEN, 2002).

Apenas a seleção não introduz novos esquemas na população – esta é uma atribuição do operador de cruzamento (*crossover*), que habilita a troca de informação estruturada, ainda que aleatória (CASTRO; ZUBEN, 2002).

O operador de cruzamento combinado com o operador de seleção pode destruir ou criar esquemas, porém a probabilidade de um esquema sobreviver ao *crossover* é maior se esse esquema for pequeno (MITCHELL, 1996). Esse aspecto foi formalizado pelo Teorema dos Esquemas. Todavia, acredita-se que o operador de cruzamento é o principal operador de um algoritmo genético, já que recombina instâncias de bons esquemas para formar instâncias de esquemas melhores ou igualmente bons aos esquemas anteriores (MITCHELL, 1996). Baseado na afirmação anterior introduz-se outro Teorema – o Teorema dos Blocos Construtivos: um algoritmo genético busca desempenho quase-ótimo através da

justaposição de esquemas curtos, de baixa ordem e alto desempenho, chamados de *blocos construtivos* (MITCHELL, 1996 apud GOLDBERG, 1989).

O Teorema dos Esquemas e o Teorema dos Blocos Construtivos tratam dos operadores de seleção e cruzamento (MITCHELL, 1996). O operador de mutação foi proposto por Holland, em 1975, para prevenir a baixa diversidade de uma posição de um esquema, ou seja, evitar casos onde, por exemplo, inicialmente todos os indivíduos começassem com 1 na primeira posição de sua cadeia de bits – a mutação seria o único meio de se obter indivíduos que possuíssem o 0 no início de sua cadeia.

3.5 Operadores Genéticos

3.4.1 Seleção de Pais

Este operador seleciona cromossomos da população para reprodução e favorece os cromossomos com maior *fitness*, ou seja, quanto melhor a avaliação do cromossomo, feita pela a função de avaliação, mais vezes este cromossomo pode ser selecionado para se reproduzir (MITCHELL, 1996).

Este método simula o mecanismo de seleção natural, onde os pais mais capazes geram mais filhos e pais menos aptos podem gerar menos descendentes (LINDEN, 2006). É importante notar que os indivíduos menos aptos não podem ser totalmente descartados, a fim de evitar a convergência genética – onde os indivíduos se tornam cada vez mais semelhantes, o que destrói a diversidade da população, comprometendo a evolução.

Existem várias maneiras de se implementar este método, algumas das quais são descritas a seguir.

ROLETA

Neste método é criada uma roleta, onde cada cromossomo recebe uma parte proporcional ao seu *fitness* em relação à soma total dos *fitness* de todos os cromossomos da população.

Como exemplo, é mostrado a seguir uma tabela com alguns indivíduos fictícios e a representação da roleta para estes indivíduos:

Tabela 3.1: Grupo de indivíduos, seus respectivos *fitness* e parcela na roleta (LINDEN, 2006)

Indivíduo	*Fitness*	Pedaço da Roleta (%)	Pedaço da roleta (°)
0001	1	1.61	5.8
0011	9	14.51	52.2
0100	16	25.81	92.9
0110	36	58.07	209.1
Total	*62*	*100*	*360.0*

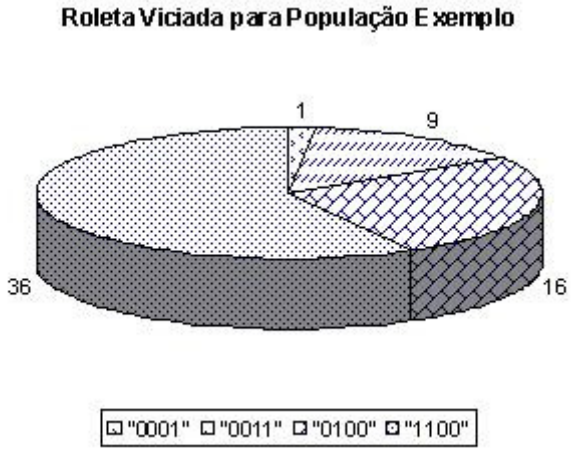

Figura 3.5: Roleta Viciada para a população exemplo da tabela 3.1 (LINDEN, 2006)

A representação computacional da roleta ilustrada acima é dada por um algoritmo que não permite que haja indivíduos com avaliação negativa ou igual a zero, já que nunca seriam selecionados, pois não possuiriam parcela alguma da roleta. A lógica do algoritmo de implementação da roleta é descrito pela listagem 3.2.

Listagem 3.2: Pseudocódigo do método de seleção da roleta (LINDEN, 2006)

```
(a) Some todas as avaliações para uma variável soma
(b) Ordene todos os indivíduos em ordem crescente de avaliação (opcional)
(c) Selecione um número s entre 0 e soma (não incluídos)
(d) i = 1
(e) aux = avaliação do indivíduo
(f) enquanto aux < s
(g)       i = i + 1
(h)       aux = aux + avaliação do indivíduo i
(i) fim enquanto
```

TORNEIO

Neste método são selecionados *N* indivíduos da população. Dentre os indivíduos selecionados, o mais apto é selecionado para a reprodução. A quantidade *N* é chamada de tamanho do torneio, e é um parâmetro definido pelo usuário. Geralmente quanto maior o valor de *N* maior a probabilidade de convergência genética, resultante da extinção dos indivíduos menos aptos da população (CARVALHO, 2005).

3.4.2 Recombinação ou Cruzamento (*Crossover*)

O processo de criação de um novo cromossomo através da combinação de um ou mais cromossomos de alta performance (cromossomo que obteve uma alta nota da função de avaliação) é conhecido como recombinação ou cruzamento (COX, 2005). A principal função deste operador é assegurar a troca de material genético entre dois indivíduos chamados pais, combinando informações de maneira que haja uma probabilidade razoável dos indivíduos resultantes deste cruzamento sejam melhores que os pais (PAPPA, 2002).

Graças ao operador de recombinação e a função de avaliação os algoritmos genéticos são classificados como uma busca dirigida, já que utiliza a seleção para determinar as áreas mais promissoras de pesquisa e a recombinação para combiná-las de modo a gerar soluções mais aptas para resolução do problema em questão (LINDEN, 2006).

No cruzamento de um ponto, são selecionados dois pais utilizando um método de seleção. É escolhido, randomicamente, um ponto de corte – uma posição entre dois genes

de um cromossomo. Então os pais são separados em duas partes, uma à direita e outra à esquerda do ponto de corte. O primeiro filho é obtido concatenando-se a parte que se encontra à esquerda do ponto de corte do primeiro pai à parte que se encontra à direita do segundo pai. O segundo filho, é obtido concatenando-se a parte que se encontra à esquerda do segundo pai à parte que se encontra à direita do ponto de corte do primeiro pai.

Já em outra forma de recombinação chamada *crossover* uniforme cada gene possui igual probabilidade *P* de ser trocado independente de sua posição. O *crossover* uniforme dá origem a apenas um novo indivíduo, enquanto que o *crossover* de um ponto resulta em dois novos indivíduos.

A recombinação uniforme é mais poderosa para algoritmos genéticos que utilizam a representação cromossomial binária, já que criam soluções que os outros tipos de recombinação, como o de um ponto, podem criar, além de criar combinações que seriam impossíveis em outros tipos de recombinação. Porém este é o tipo de *crossover* que mais facilmente quebra esquemas interessantes (LINDEN, 2006). A figura 3.5 ilustra o cruzamento de um ponto e o cruzamento uniforme.

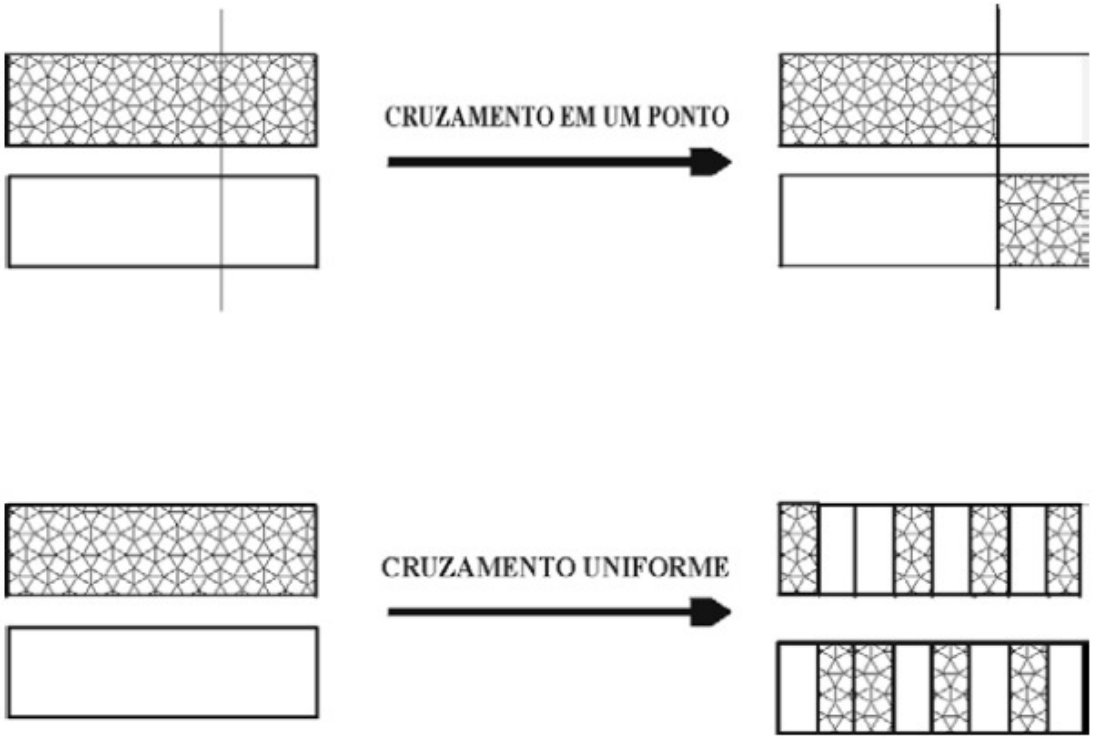

Figura 3.6: Tipos de cruzamento (*crossover*) (CARVALHO, 2005)

3.4.3 Mutação

Este operador troca randomicamente alguns *bits* de um cromossomo. Por exemplo, a seqüência 00000100 poderia ser modificada em sua segunda posição, transformando-se na seqüência 01000100. A mutação pode ocorrer em cada gene de um cromossomo com uma probabilidade informada pelo usuário, ou baseada em algum critério previamente definido. (MITCHELL, 1996).

O propósito do operador de mutação é manter a diversidade da população e assegurar que o cromossomo sempre cobrirá uma parte suficientemente grande do espaço de busca (PAPPA, 2002 apud Hinterding 2000), introduzindo material genético que não está presente em nenhum outro indivíduo da população, ao contrário do operador de *crossover* (CARVALHO, 2005).

A mutação diversifica a população e combate as regiões de mínimos e máximos locais, assegurando o surgimento de novas soluções potenciais, independentes dos cromossomos já existentes (COX, 2005).

3.6 Elitismo

A cada geração *G* de um AG, os indivíduos resultantes da geração *G-1* são todos descartados, dando lugar a novos indivíduos resultantes da aplicação dos operadores de seleção, cruzamento e/ou mutação aos indivíduos da geração *G-1*.

A estratégia elitista visa preservar indivíduos com altos valores de *fitness*, ou seja, as melhores soluções encontradas na geração corrente, por mais de uma geração, copiando-os para a geração seguinte (PAPPA, 2002). Seja uma população com *N* indivíduos. São escolhidos N_{elit} indivíduos com os melhores *fitness*. Estes indivíduos são copiados integralmente para a população da próxima geração do AG. Os outros N- N_{elit} indivíduos da nova população são gerados a partir da aplicação dos operadores genéticos citados acima. O número N_{elit} é chamado de fator de elitismo, e é um número, definido pelo usuário e geralmente pequeno (CARVALHO, 2005).

3.7 Nichos Biológicos

Simulando o processo evolutivo natural, onde cada espécie evolui em nichos ecologicamente separados e consomem os recursos oferecidos neste nicho, os métodos de *niching* são utilizados para evitar a convergência do algoritmo genético para uma população uniforme, com indivíduos semelhantes a um super indivíduo (uma solução de alta qualidade) (CARVALHO; FREITAS, 2002).

A formação de espécies permite primeiramente que vários disjuntos e diversos conceitos sejam aprendidos simultaneamente, além de permitir que os recursos computacionais sejam efetivamente explorados, evitando replicações desnecessárias e redundâncias (GIORDANA; NERI, 1995)

Os métodos de nicho (*niching*) não permitem que a população perca a sua diversidade, habilitando os AGs a localizar diferentes soluções com boa qualidade e mantê-las numa mesma subpopulação (CARVALHO, 2005). A fundamentação para a introdução deste método no AG vem da observação dos ecossistemas biológicos – na natureza, se uma determinada espécie satura um determinado ambiente, os indivíduos daquela espécie são forçados a compartilhar os recursos disponíveis, o que nos faz concluir que a necessidade de compartilhamento é uma conseqüência natural de ambiente contendo superpopulações e conflitos (CARVALHO, 2005).

As técnicas de *niching* são importantes para o sucesso dos AGs aplicados a tarefas como a classificação, o aprendizado de máquina, a otimização de funções multimodais, a otimização de funções multiobjetivos e simulação de sistemas adaptativos e complexos (CARVALHO; FREITAS, 2002). A seguir são descritos alguns métodos de niching, a fim de ilustrar como esta técnica é introduzida aos algoritmos genéticos.

3.6.1 *Fitness Sharing*

Neste método, a função de avaliação é utilizada para forçar a diversidade dos indivíduos. No *Fitness Sharing*, na avaliação de um indivíduo, não é considerado apenas o seu *fitness* individual, também é utilizado o *fitness* de todos os indivíduos que estão próximos no espaço de busca (CARVALHO, 2005). O cálculo do *fitness* de um indivíduo *I* para

o método de *fitness sharing* é realizado dividindo-se o fitness de *I* pelo valor de sua contagem de nicho (*niche count*). Esta função de nicho, para cada indivíduo *I* é o valor da soma dos valores da função de *sharing* entre ele e cada outro indivíduo na população (incluindo ele mesmo). A função de *sharing* avalia a igualdade entre dois indivíduos, retornando 1 se estes forem idênticos e 0 se eles ultrapassam um limiar de diferença previamente definido. Os cálculos demandados para a implementação deste método o tornam computacionalmente caro, já que em uma população com *N* indivíduos, N^2 cálculos devem ser feitos para determinar o *fitness* de um individuo (CARVALHO, 2005). Uma vantagem deste método é a sua capacidade de espalhar a população em múltiplos nichos.

3.6.2 COGIN (*COvered-based Genetic INduction*)

O COGIN é um sistema de aprendizado que utiliza um método de niching implícito (CARVALHO; FREITAS, 2002), utilizando um algoritmo de indução de regras baseado na competição (CARVALHO, 2005).

O COGIN funciona criando novos indivíduos (regras) candidatos através de cruzamentos utilizando os indivíduos presentes na própria população. Este cruzamento é feito utilizando cruzamento de um ponto. Após a etapa do cruzamento os indivíduos são ordenados de maneira crescente pelo valor *p* retornado pela função de avaliação para este indivíduo. Então é tomada a primeira regra (indivíduo de maior *fitness*) e os exemplos do conjunto de treinamento que atendem a esta regra são escolhidos e eliminados. Este processo se repete até que o conjunto de treinamento não possua mais nenhum exemplo a ser coberto, então as regras restantes são descartadas, já que sua presença na população de indivíduos é desnecessária já que todos os exemplos já foram cobertos pelas regras geradas, que substituem as regras existentes. Pode-se notar que o tamanho da população de indivíduos é variável, dependendo exclusivamente da quantidade de regras necessárias para cobrir todo o espaço do conjunto de treinamento, e esta é uma grande vantagem deste método, pois o número de regras se adapta dinamicamente aos dados (CARVALHO, 2005 apud GREENE, D.P.; SMITH, S, 1993). A desvantagem deste método é a difícil compreensão do conjunto de regras, pois estas se apresentam a partir de um conjunto de regras ordenadas (CARVALHO; FREITAS, 2002).

3.6.3 REGAL (*RElational Genetic Algorithm Learner*)

O REGAL é um sistema distribuído, baseado em algoritmos genéticos, desenvolvido para aprendizagem de conceitos representados em lógica de primeira ordem a partir de exemplos (GIORDANA; NERI, 1995). Este sistema é baseado num operador de seleção chamado sufrágio universal (*universal suffrage*), que tem como função promover a coexistência de diferentes espécies.

O sufrágio universal utiliza uma metáfora política (FREITAS, 2007) onde os exemplos da base de treinamento podem votar apenas num indivíduo (regra) que os representa. Cada exemplo pode votar apenas uma vez. Os indivíduos mais votados são selecionados para o cruzamento. Este método é descrito com mais detalhes na seção 4.4.

4 DESCOBERTA DE REGRAS UTILIZANDO AGs

A utilização dos algoritmos genéticos para a tarefa de descoberta de regras de previsão se deve a busca global realizada por esta abordagem. Outro fator determinante para a obtenção de bons resultados na aplicação de algoritmos genéticos a esta tarefa é a boa interatividade com os atributos, que é inerente aos processos de seleção e cruzamento providos pelos algoritmos genéticos. Essa interatividade é maior que a interatividade oferecida pelos algoritmos freqüentemente utilizados para *Data Mining*, geralmente baseados na estratégia gulosa (FREITAS, 2001 apud DHAR V; PROVOST, 2000).

Neste capítulo serão discutidos os aspectos e as ferramentas utilizadas no uso de algoritmos genéticos para descoberta de regras de previsão no processo de Mineração de Dados.

4.1 Codificação do Indivíduo

Algoritmos genéticos para descoberta de regras podem ser divididos em duas grandes abordagens, que diferem entre si na maneira como elas representam as regras em um indivíduo da população de soluções.

Em geral, existem duas abordagens para a representação de indivíduos em um algoritmo genético para descoberta de regras: a abordagem de Michigan e a abordagem de Pittsburgh.

Na abordagem de Michigan, cada indivíduo representa uma única regra, enquanto que para a abordagem de Pittsburgh, cada individuo representa um conjunto de regras (FREITAS, 2001). Cada abordagem tem sua aplicação em uma etapa diferente do processo de Mineração de Dados. A abordagem de Michigan é aplicada à descoberta de regras de previsão de uma maneira mais natural, pois nesta tarefa a avaliação de cada regra é mais interessante para o sucesso no alcance do objetivo. Se for avaliada cada regra

individualmente, ao final do algoritmo genético as regras que possuem os melhores índices de previsão serão conhecidas.

Para a tarefa de classificação de dados, a abordagem mais adequada dentre as duas é a de Pittsburgh, já que nesta abordagem, são avaliados os conjuntos de regras. Os melhores conjuntos avaliados classificarão os dados de maneira mais satisfatória, o que é o objetivo neste momento.

Enquanto a abordagem de Pittsburgh oferece interação entre as regras, a abordagem de Michigan oferece interação entre os atributos, o que justifica que cada abordagem seja mais bem aplicada em tarefas diferentes. Os indivíduos gerados pela abordagem de Pittsburgh tendem a terem mais informação e a serem mais complexos (FREITAS, 2001), o que aumenta o custo computacional para o cálculo do *fitness*. Por outro lado, os indivíduos baseados na abordagem de Michigan são mais simples e menores, o que acelera o desempenho da função de avaliação.

Para este trabalho, foi escolhida a representação binária seguindo a abordagem de Michigan por facilitar a representação das regras e atributos, além da interação entre cada um deles durante o processo de execução do algoritmo.

4.2 Operador de Introdução de Novos Indivíduos (*Seeding*)

O operador de *seeding* introduz na população um novo indivíduo. O diferencial deste operador é que o novo indivíduo gerado cobre um exemplo pertencente a base de treinamento. Este operador age como uma função, que, dado um exemplo pertencente a base de treinamento retorna um indivíduo que cobre este exemplo (GIORDANA; NERI, 1995). A listagem 4.1 Descreve o funcionamento deste operador.

Listagem 4.1: Pseudocódigo do operador de *seeding* (GIORDANA; NERI, 1995).

```
Seja e um exemplo pertencente ao conjunto de treinamento E
Gere uma cadeia aleatória de bits definindo um indivíduo K
Transforme em 1 o menor número de bits em K para que K cubra e
Retorne (K)
```

4.3 Função de Avaliação para Descoberta de Regras de Classificação

A função de avaliação deve traduzir o problema a ser resolvido para que possa ser compreendido e tratado por um computador (LINDEN, 2006). Para a descoberta de regras de classificação, a função de avaliação deve ser capaz de medir as seguintes características: a acurácia preditiva da regra, ou seja, o poder desta regra de prever comportamentos baseada nas informações cedidas pelo conjunto de treinamento; o quanto esta regra é compreensível ao usuário, pois regras que não são claras ou que são muito complexas serão sumariamente descartadas pelo usuário; e o quanto esta regra é interessante, sendo esta a característica mais difícil de ser medida, por ser extremamente subjetiva (FREITAS, 2001).

Para um AG que segue a abordagem de Michigan, onde cada indivíduo representa apenas uma regra, seja esta regra representada pela forma SE A ENTÃO C, onde A representa a parte antecedente da regra e C a parte conseqüente da regra – o atributo objetivo. Uma maneira simples de medir a acurácia preditiva (AC) de uma regra é definida por:

$$AC = \frac{|\, A \,\&\, C|}{|\, A \,|}, \tag{4.1}$$

onde $|\, A \,|$ é o número de exemplos da base de dados que satisfazem todas as condições antecedentes representadas por A e $|\, A \,\&\, C|$ representada todos os exemplos que atendem a parte antecedente e conseqüente da regra (exemplos que atendem a A e a C) (FREITAS, 2001). Por exemplo, se uma regra cobre 10 exemplos ($|\, A \,| = 10$), dos quais apenas 8 possuem o mesmo valor de atributo objetivo (classe) que esta regra ($|\, A \,\&\, C| = 8$), tem-se que a acurácia preditiva desta regra é igual a $\frac{8}{10} = 80\ \%$.

A acurácia preditiva de uma regra de classificação pode ser resumida a uma matriz 2x2 chamada de matriz de confusão. Seu conteúdo é representado a partir de quatro conceitos que são observados ao utilizarmos uma regra para classificar um exemplo da base

de dados a partir da classe que foi prevista pela regra e da classe efetiva deste exemplo, e são apresentados a seguir:

- Verdadeiros Positivos (VP) – denota o número de exemplos que atendem a parte antecedente da regra (a parte A) e a parte conseqüente da regra (a parte representada por C);

- Falsos Positivos (FP) – denota o número de exemplos que atendem a parte antecedente da regra (a parte A) e não atendem a parte conseqüente da regra (a parte representada por C);

- Falsos Negativos (FN) – denota o número de exemplos que não atendem a parte antecedente da regra (a parte A), porém atendem a parte conseqüente da regra (a parte representada por C);

- Verdadeiros Negativos (VN) – denota o número de exemplos que não atendem a parte antecedente da regra (a parte A) e não atendem a parte conseqüente da regra (a parte representada por C);

Assim, a forma de uma matriz de confusão é apresentada a seguir:

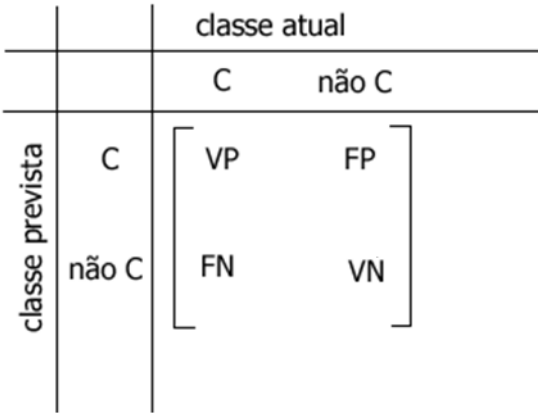

Figura 4.1: Matriz de Confusão para uma Regra de Classificação (FREITAS, 2001).

Outra fórmula para medir a acurácia preditiva de cada regra, baseando-se nos conceitos apresentados acima é representada pela fórmula:

$$AC = \frac{|VP|}{|VP+FP|}, \tag{4.2}$$

Outro conceito importante para a avaliação de regras de classificação é o conceito da completude, que representa o número de exemplos da base de dados que possuem o mesmo valor do atributo objetivo que a regra a ser avaliada e que são cobertos pela regra antecedente. A completude *Comp* de uma regra pode ser avaliada através da fórmula:

$$Comp = \frac{|VP|}{|VP+FN|}. \quad (4.3)$$

Tendo em vista que as características citadas acima, acurácia preditiva e completude, são as características desejáveis a regras de classificação, podemos então utilizar a seguinte função de avaliação, onde o valor do *fitness* é obtido da seguinte forma:

$$Fitness = AC\ x\ Comp\ , \quad (4.4)$$

e que foi utilizada para a avaliação das regras neste trabalho.

Vale salientar que, durante a execução do algoritmo genético para extração de regras de classificação, indivíduos que possuem acurácia preditiva de 100% não são interessantes, por representarem idiossincrasias particulares ao conjunto de dados utilizado para treinamento das regras. A função de *fitness* utilizada neste trabalho não avalia a simplicidade de uma regra.

4.4 Métodos para Manter a Diversidade de Regras

Como discutido na seção 3.5, existem alguns métodos de criação de nichos que visam à manutenção da diversidade entre os indivíduos de uma população em um algoritmo genético.

Para a manutenção da diversidade de indivíduos em um algoritmo genético para a descoberta de regras de classificação, existe ainda outra maneira para se manter a diversidade das regras, evitando a convergência para um indivíduo: a Cobertura Seqüencial (*Sequential Covering*). Neste método, a cada execução do AG apenas uma regra é selecionada – a regra com melhor adaptação. Após a execução do AG, a regra mais apta é armazenada e todos os exemplos do conjunto de treinamento que são cobertos por esta regra são descartados. O AG é executado *n* vezes, onde *n* é a quantidade de vezes

necessárias para que haja regras suficientes para cobrir todo o conjunto de treinamento, ou seja, até que o conjunto de treinamento esteja vazio. Portanto, o uso deste método de seleção implementa uma forma de niching, fomentando a evolução de diferentes regras que cobrem diferentes partes da base de dados (FREITAS, 2007). No sufrágio universal, o método de seleção favorece os indivíduos que cobrem mais exemplos, pois eles ocorrem várias vezes na roleta, e os indivíduos mais adaptados (maior *fitness* total), pois possuem maior probabilidade de serem selecionados pela roleta.

Para este trabalho foi utilizado um método baseado em nichos para a manutenção da diversidade das regras: o sufrágio universal. Este método foi escolhido por evitar a convergência da população em torno de um super-indivíduo que foi obtida quando foi utilizado apenas o método de seleção da roleta. A listagem 4.1 exibe o pseudocódigo do funcionamento do sufrágio universal.

Listagem 4.2: Pseudocódigo do método de sufrágio universal (GIORDANA; NERI, 1995).

```
B(t) = Ø
Selecione aleatoriamente, com reposição g * M exemplos de E
Para cada exemplo K selecionado faça
        Para cada individuo I da população faça
            Se existe um indivíduo que cobre o K
                    então armazene este indivíduo como um candidato
            Senão crie um novo indivíduo que cubra este exemplo e adicione a B(t)
        Fim
        Utilize o método da roleta para escolher apenas um indivíduo que receberá o voto
        deste exemplo e o adicione a B(t)
Fim
```

A cada geração do algoritmo genético é selecionado, de maneira aleatória, um subconjunto de exemplos do conjunto de treinamento E. Este conjunto de treinamento é um subconjunto da base de dados utilizada para se obter as regras de predição. Esta seleção é realizada com reposição dos exemplos para que, casos onde o número de exemplos seja menor que o número de indivíduos M da população (E < M) não haja mais exemplos a serem escolhidos (GIORDANA; NERI, 1995). Para cada exemplo selecionado, são escolhidos os candidatos a receberem o voto deste exemplo, ou seja, as regras que cobrem este exemplo

e que são agora passíveis de serem votados por este exemplo, esses candidatos são armazenados. Depois de se conhecer e armazenar todos os candidatos, o indivíduo votado é escolhido utilizando o método de seleção da **roleta**, onde é selecionado um indivíduo a partir de um subconjunto de indivíduos com probabilidade diretamente proporcional a seu *fitness*, assim a probabilidade de ser escolhido um indivíduo de alto *fitness* é grande, porém indivíduos de baixo f*itness* não estão totalmente descartados, o que é importante para a manutenção da diversidade da população de um AG. Apenas os exemplos pertencentes ao conjunto B(t) são selecionados para cruzamento, e é esse aspecto que garante que apenas formulas que cobrem o mesmo exemplo sejam competidoras entre si. A introdução do método de seleção da roleta, que escolhe o indivíduo que será votado pelo exemplo, faz com que a probabilidade de que os exemplos votem nos melhores indivíduos seja proporcional ao *fitness* dos indivíduos que cobrem estes exemplos. A função de fitness a ser utilizada não precisa necessariamente levar em consideração a completude de cada regra, já que apenas indivíduos que cobrem pelo menos um exemplo são selecionados quando é utilizado o operador de seleção sufrágio universal (RAS; ZEMANKOVA, 1994), já que estes indivíduos devem ser votados por pelo menos um exemplo. Porém, para este trabalho foi considerada a completude de uma regra na função de *fitness* para que regras que cubram mais exemplos que outras tenham maior probabilidade de serem votadas.

4.5 Operador de Cruzamento (*Crossover*)

O operador de cruzamento utilizado para este trabalho foi o *crossover* uniforme. Este tipo de *crossover* foi escolhido por ser capaz de combinar todo e qualquer esquema existente na população (LINDEN, 2006).

Existe também outro tipo de *crossover* específico para a tarefa de descoberta de regras de classificação: o *crossover* de generalização/especialização (*generalizing/specializing crossover*). A idéia básica deste operador é generalizar ou especializar uma dada regra, ou seja, aumentar o número de exemplos que são cobertos pela regra se este número for considerado pequeno, ou aumentar o número de exemplos que são cobertos por esta regra se este for considerado grande.

Como exemplo, consideremos que um algoritmo evolucionário, seguindo a codificação de Michigan. Então, o *crossover* de generalização/especialização pode ser implementado como o OU lógico ou o E lógico (FREITAS, 2001), respectivamente, como mostrado na figura 4.2.

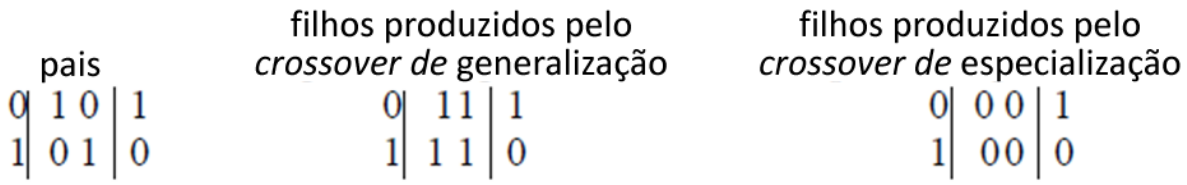

Figura 4.2: Exemplo de *crossover* de generalização/especialização (FREITAS, 2001)

4.6 Operador de Generalização e Especialização

A generalização ou especialização de regras pode também ser realizada por um operador independente do operador de cruzamento, chamado operador de generalização/especialização.

Para ilustrar o uso deste operador, consideremos a seguinte regra, onde as condições são ligadas por um E lógico (FREITAS, 2001):

$$(idade > 25)\ (estado_{civil} = solteiro) \tag{4.5}$$

Esta regra pode ser generalizada, utilizando a mutação para subtrair certo valor de 25, assim em 4.5 haverá uma cobertura de mais exemplos. Um exemplo pode ser ilustrado através em 4.6 (FREITAS, 2001):

$$(idade > 21)\ (estado_{civil} = solteiro) \tag{4.6}$$

Outra maneira de generalizar uma regra é apagar alguma das condições desta regra. Para realizar a especialização de uma regra, pode-se adicionar certo valor a 25, assim diminuiria o intervalo coberto e assim o número de exemplos cobertos por este indivíduo.

Para este trabalho este operador não foi utilizado e a cobertura dos exemplos é garantida pela função de avaliação e pelo método de sufrágio universal.

5 IMPLEMENTAÇÃO

O presente capítulo descreve e discute o algoritmo implementado, baseado em técnicas da Programação Genética, para a ferramenta EPT - *Explorer Patterns Tool*, ferramenta de Mineração de Dados desenvolvida neste trabalho. Esta ferramenta é a resultante da incorporação de algoritmos genéticos para extração de regras de predição à ferramenta EFT (*Explorer Fuzzy Tree*). A ferramenta EFT foi desenvolvida em SOUZA, 2008 para oferecer ao seu usuário parâmetros de experimentação que permitam aumentar o poder preditivo da árvore clássica através da sua fuzzyficação (SOUZA, 2008).

O EPT oferece um ambiente de experimentos com algoritmos de classificação baseados em árvores de decisão nebulosas e clássicas sobre qualquer base de dados com atributos contínuos (SOUZA, 2008) assim como experimentos com algoritmos genéticos para extração de regras de predição sobre qualquer base de dados com atributos contínuos ou categóricos, além de um ambiente onde é possível comparar esses dois paradigmas utilizados para a Mineração de Dados.

Nas próximas seções serão descritas as principais características da ferramenta: sua arquitetura, o algoritmo implementado. Informações acerca do uso da ferramenta são explanadas no apêndice A.

5.1 Arquitetura da Ferramenta

A ferramenta *Explorer Patterns Tool* é baseada no modelo de arquitetura de três camadas e foi totalmente desenvolvida utilizando a linguagem Java, e é compatível com o Java versão 6.

A modelagem em três camadas foi utilizada para o desenvolvimento desta ferramenta. Neste modelo, é feita uma separação das três camadas de software: a camada de dados é separada da camada de interface com o usuário e da camada de negócios. Esta

separação tem por fim promover a independência entre essas três camadas e assim facilitar a manutenção e evolução da ferramenta incentivando novas contribuições (SOUZA, 2008). A seguir são as três camadas do EPT são descritas.

- Camada de dados – esta camada realiza o acesso e a manipulação dos dados carregados a partir das bases de dados e que serão utilizados para a mineração de dados.

- Camada de negócio – esta camada implementa toda a lógica do EPT, manipulando os dados recebidos da camada de dados, aplicando os algoritmos desenvolvidos para esta ferramenta, de acordo com a escolha do usuário, e enviando os resultados à camada visualizada pelo usuário: a camada de interface.

- Camada de interface – esta é a camada responsável pela apresentação dos resultados obtidos pelo EPT ao usuário. Detalhes sobre esta camada são explanados em 5.3.

5.2 Algoritmo Implementado

Neste trabalho um algoritmo genético para descoberta de regras de predição foi incorporado à ferramenta EFT (*Explorer Fuzzy Tree*).

O algoritmo genético implementado para este trabalho utiliza a codificação de indivíduo binária (formada por 0 e 1), baseada na abordagem de Michigan, onde cada indivíduo representa uma única regra.

Os operadores genéticos utilizados foram os operadores de *seeding*, seleção, mutação e crossover. A seguir é discutido o uso de cada um desses operadores.

A discretização dos valores contínuos foi feita utilizando o método de intervalos iguais (*equal-width discretization*). Este método divide o conjunto de valores reais de um atributo em *N* intervalos de tamanhos iguais, onde seja o *A* e *B* os limites inferior e superior

dos valores do atributo a ser discretizado, tamanho deste intervalo é dado por (*B-A*) / *N*, onde *N* é a quantidade de intervalos que devem ser gerados pela discretização. Na ferramenta EPT este valor é informado pelo usuário. Este método foi escolhido por ser um método simples de discretização.

O algoritmo genético implementado neste trabalho, pode trabalhar com dados contínuos ou categóricos na mesma base de dados, inclusive podendo utilizar atributos categóricos não apenas como atributos objetivo.

5.2.1 Inicialização da População

Para a inicialização da população do algoritmo genético implementado foi utilizado o operador de introdução de novos indivíduos (*seeding*) discutido no capítulo 4. Este operador foi escolhido para que seja garantido que os indivíduos da população do AG implementado cubram pelo menos um exemplo do conjunto de treinamento, além de garantir a presença de um nicho ecológico desde a primeira execução do algoritmo. A figura 5.1 mostra o diagrama de seqüência da função que inicializa a população do algoritmo genético implementado para este trabalho.

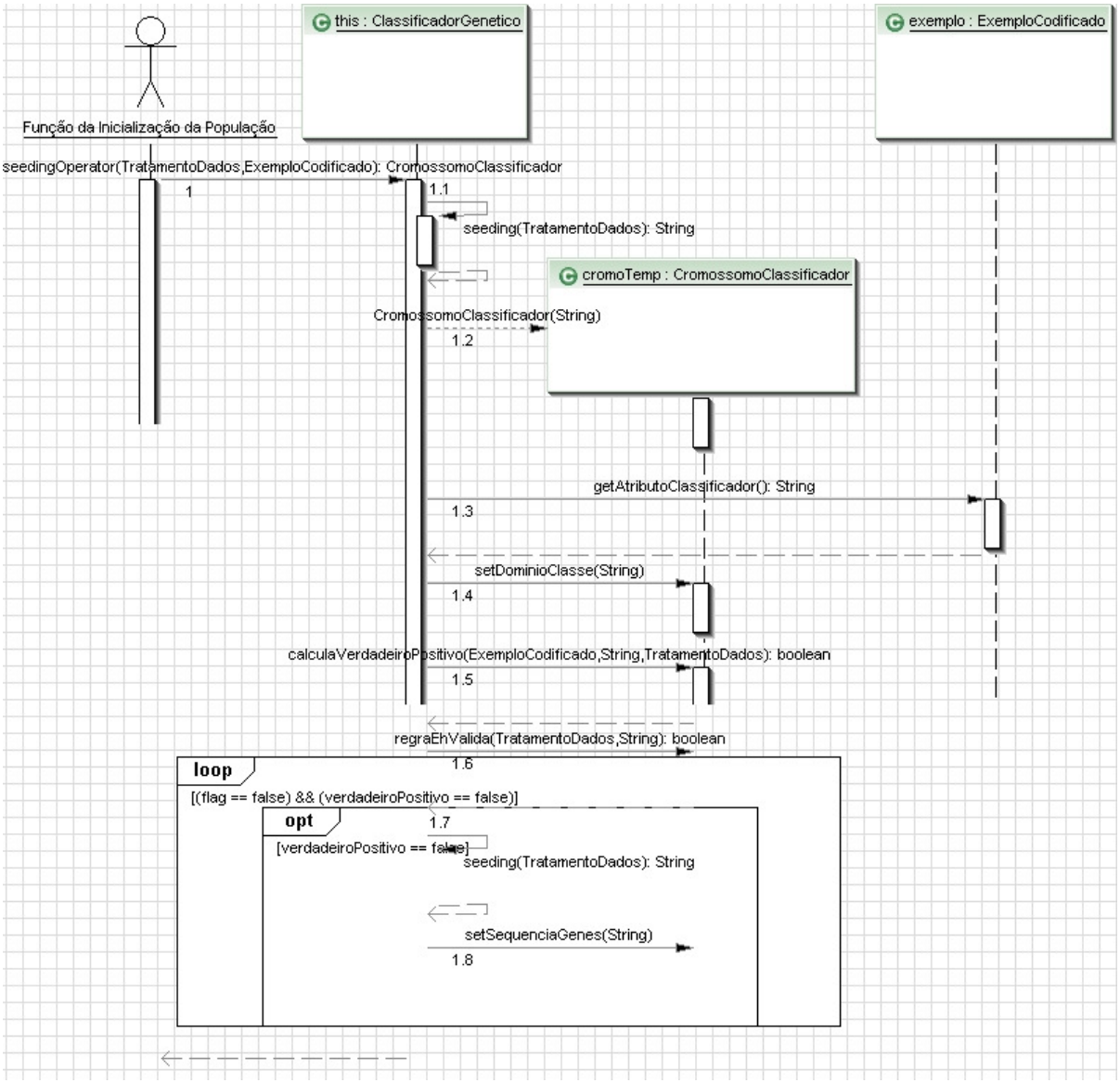

Figura 5.1: Diagrama de Classe da Função de Seeding

5.2.2 Seleção de Pais

A seleção de pais do AG objeto deste trabalho é realizada utilizando-se o operador de sufrágio universal discutido nos capítulos 3 e 4. Este operador foi escolhido para esta tarefa, por manter a diversidade da população através de nichos, e utilizar os próprios exemplos de treinamento do AG para construir esses nichos. Dessa maneira as regras geradas refletem as informações da base de dados, representadas no AG pelo conjunto de treinamento. A figura 5.2 mostra o diagrama de seqüência da função que implementa o operador de sufrágio

universal. Esta função retorna uma lista contendo os elementos que foram selecionados utilizando os princípios deste método.

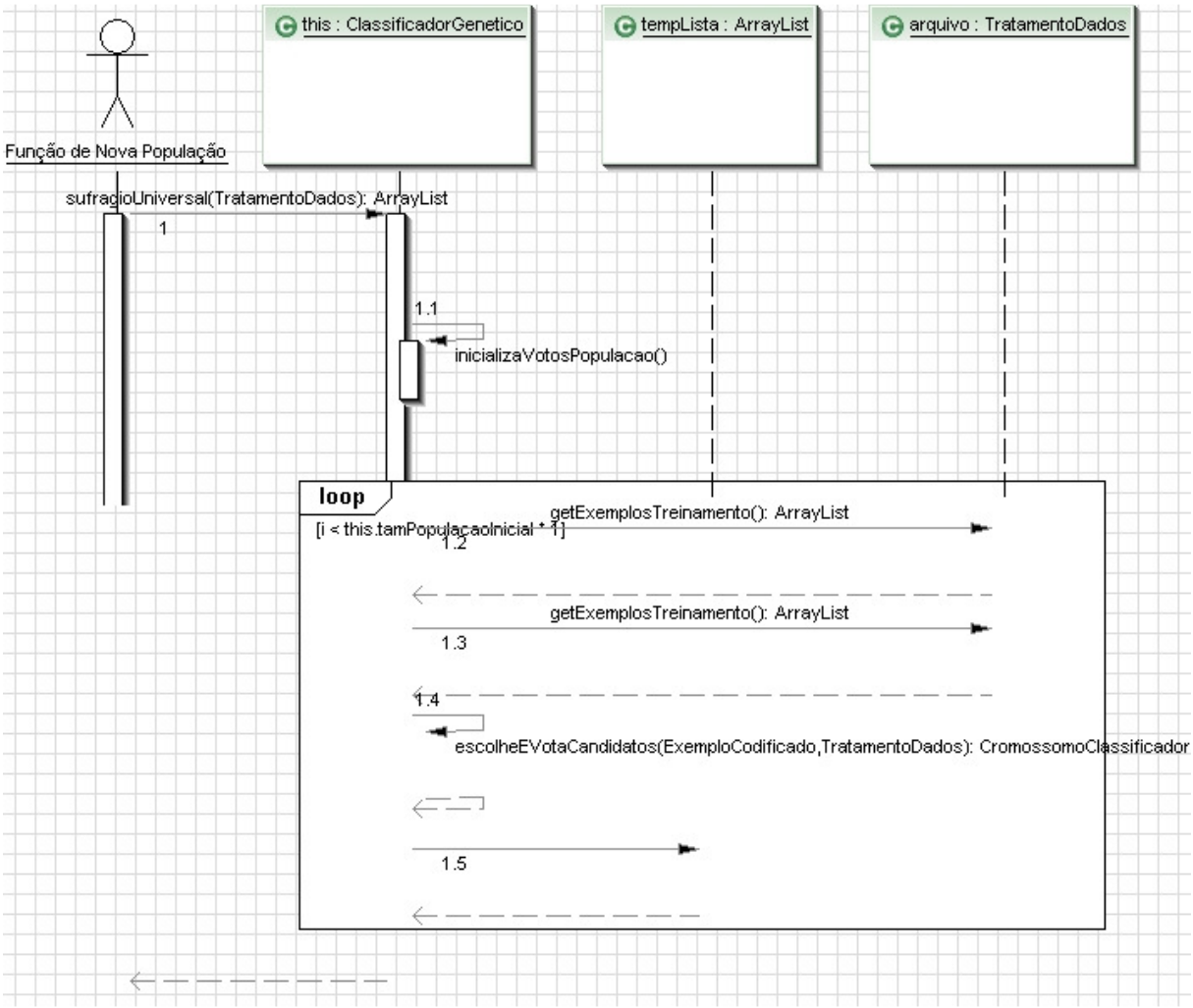

Figura 5.2: Diagrama de Classe da Função do Operador de Sufrágio Universal

5.2.3 Cruzamento, Mutação e Elitismo: Evolução dos Indivíduos

Para este trabalho foram utilizados os métodos de cruzamento, mutação e elitismo para a evolução dos indivíduos e formação da nova população do AG. O método de elitismo, discutido no capítulo 3 foi utilizado com fator de elitismo igual a 1.

Os operadores de mutação e o cruzamento uniforme, discutidos no capítulo 3, foram utilizados para a criação de novos indivíduos. Dois indivíduos são selecionados, a partir do operador de sufrágio universal e são submetidos ao operador de cruzamento uniforme,

dando origem a um novo indivíduo *K*. É aplicado neste novo indivíduo *K* o operador de mutação simples, que pode mudar cada um dos bits deste novo indivíduo com uma probabilidade de 1/L onde L é o tamanho da cadeia de genes dos indivíduos deste AG. Esta probabilidade é baseada no trabalho de Bremermann (LINDEN, 2006 apud Michalewicz, 2002), que defende que para cromossomos binários esta seria uma tava de mutação ótima. O código utilizado para implementação da mutação e do *crossover* uniforme são os mesmos códigos utilizados por (LINDEN, 2006). O diagrama de classe das funções de mutação e cruzamento e são exibidos nas figuras 5.3 e 5.4 respectivamente.

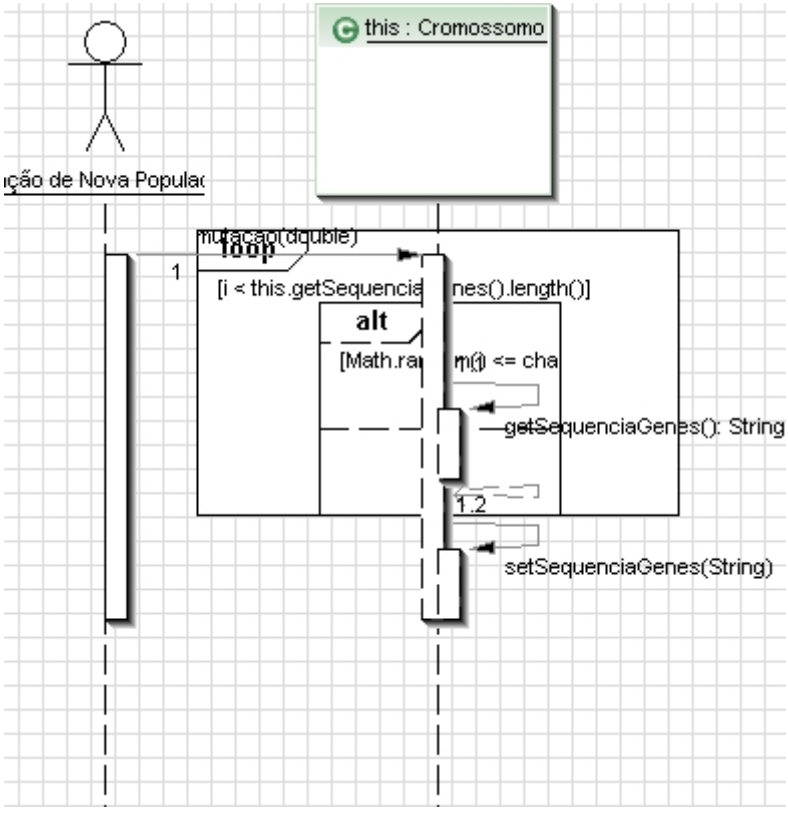

Figura 5.3: Diagrama de Classe da Função do Operador de Mutação

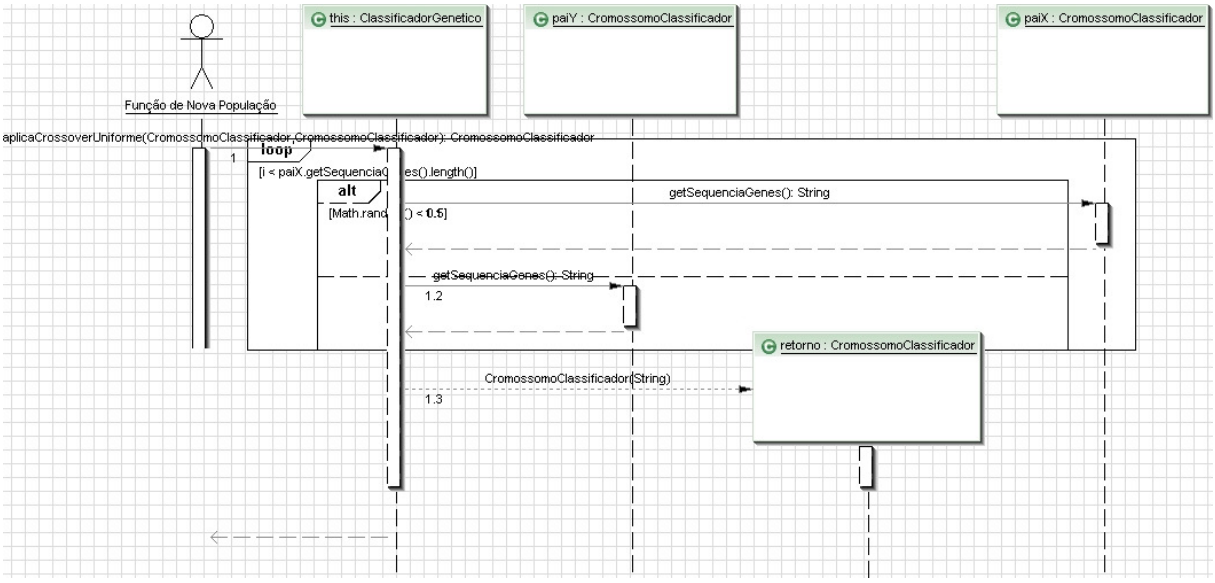

Figura 5.4: Diagrama de Classe da Função do Operador de Cruzamento Uniforme

5.2.4 Cálculo da Função de Avaliação

Para este trabalho, foi utilizada a função de avaliação descrita em 4.3 (equação 4.4), baseada nos valores de **verdadeiros positivos**, **falsos negativos** e **falsos positivos**, calculados para este indivíduo. Foi utilizado em conjunto com esta função de fitness o método descrito em (FREITAS, 2001), onde o valor para o atributo objetivo é escolhido baseado no valor do *fitness* para o indivíduo, caso seja atribuído um valor *p* pertencente ao domínio *D* do atributo objetivo. Para cada indivíduo, o cálculo do fitness é feito *n* vezes, onde *n* é o tamanho do conjunto *D*. É escolhido então o valor *p* que maximiza a função de *fitness* para o indivíduo avaliado. A figura 5.5 mostra o diagrama de classes da função Java que implementa este método.

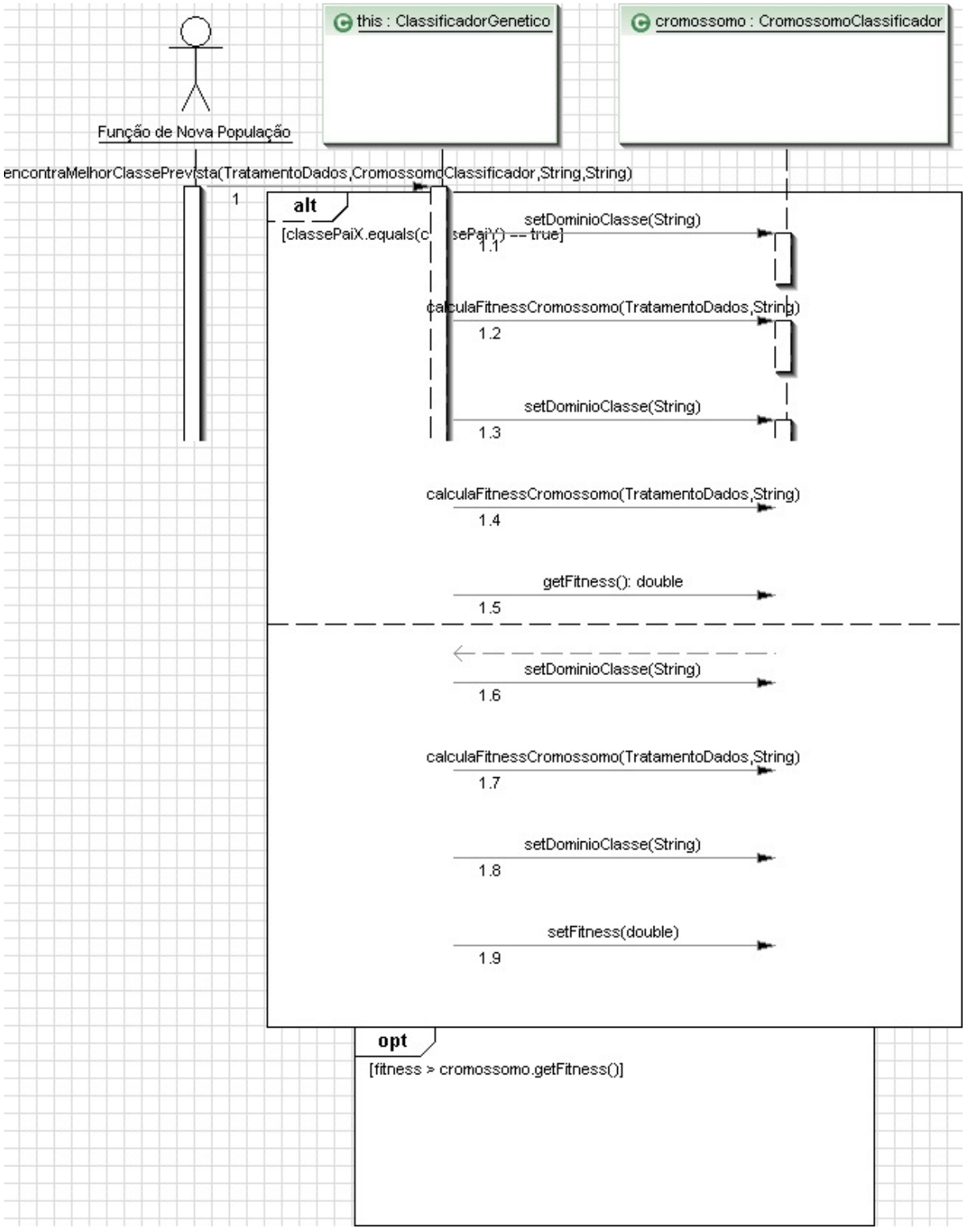

Figura 5.5: Diagrama de Classe da Função de *Fitness*

5.3 Experimentos

O *Explorer Patterns Tool* possibilita a realização de experimentos a cerca de reconhecimento de padrões sobre bases de dados utilizando regras de classificação extraídas por um algoritmo genético e árvores de decisão *fuzzy* e clássicas, ou ainda com ambos os paradigmas. Existe uma tela para cada paradigma citado anteriormente, onde o usuário pode informar os parâmetros necessários para a realização do respectivo experimento. As figuras 5.6, 5.7 e 5.8 mostram, respectivamente, a realização de experimentos para o paradigma nebuloso, genético e os experimentos que comparam esses dois paradigmas.

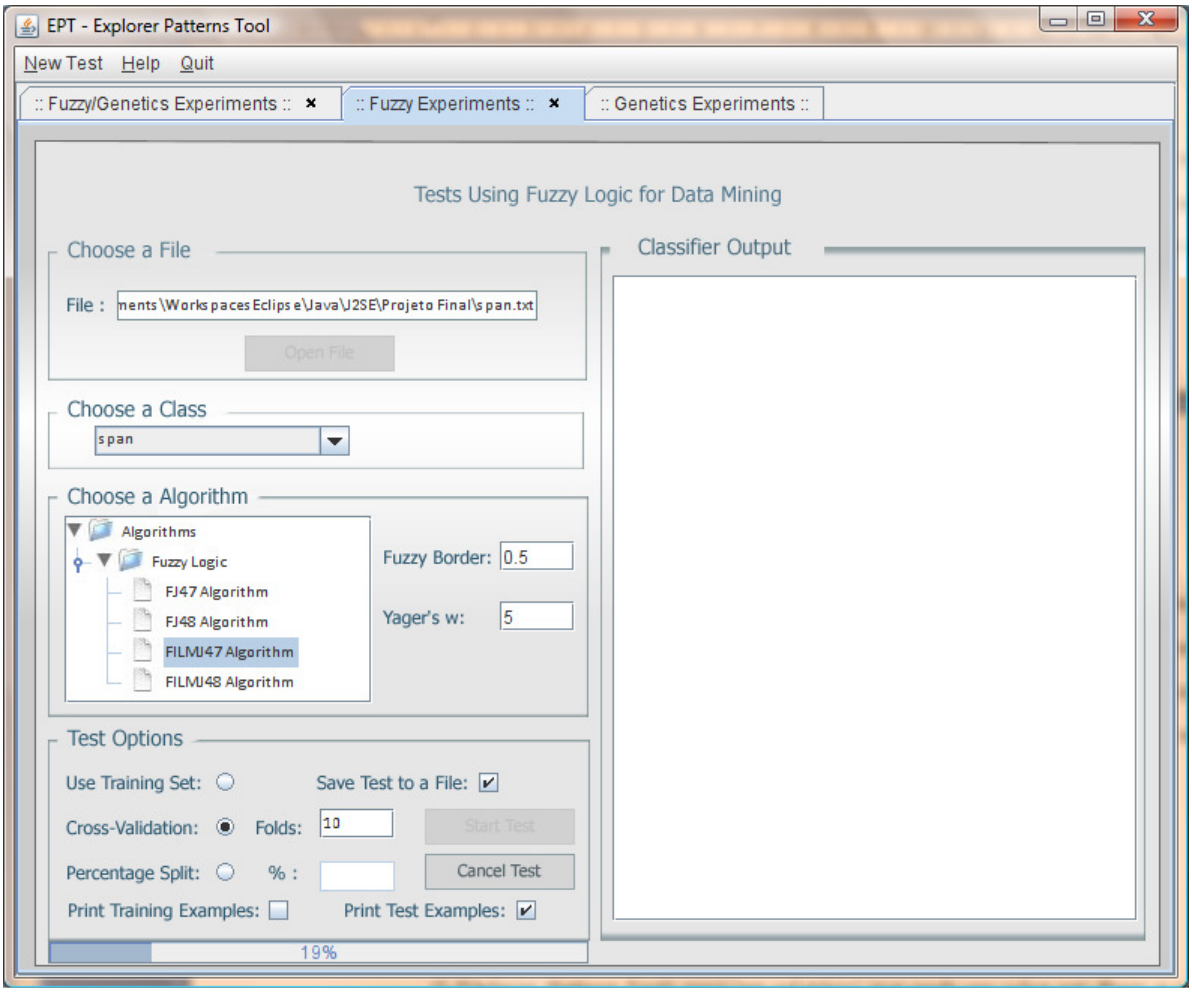

Figura 5.6: Execução de experimento utilizando lógica nebulosa.

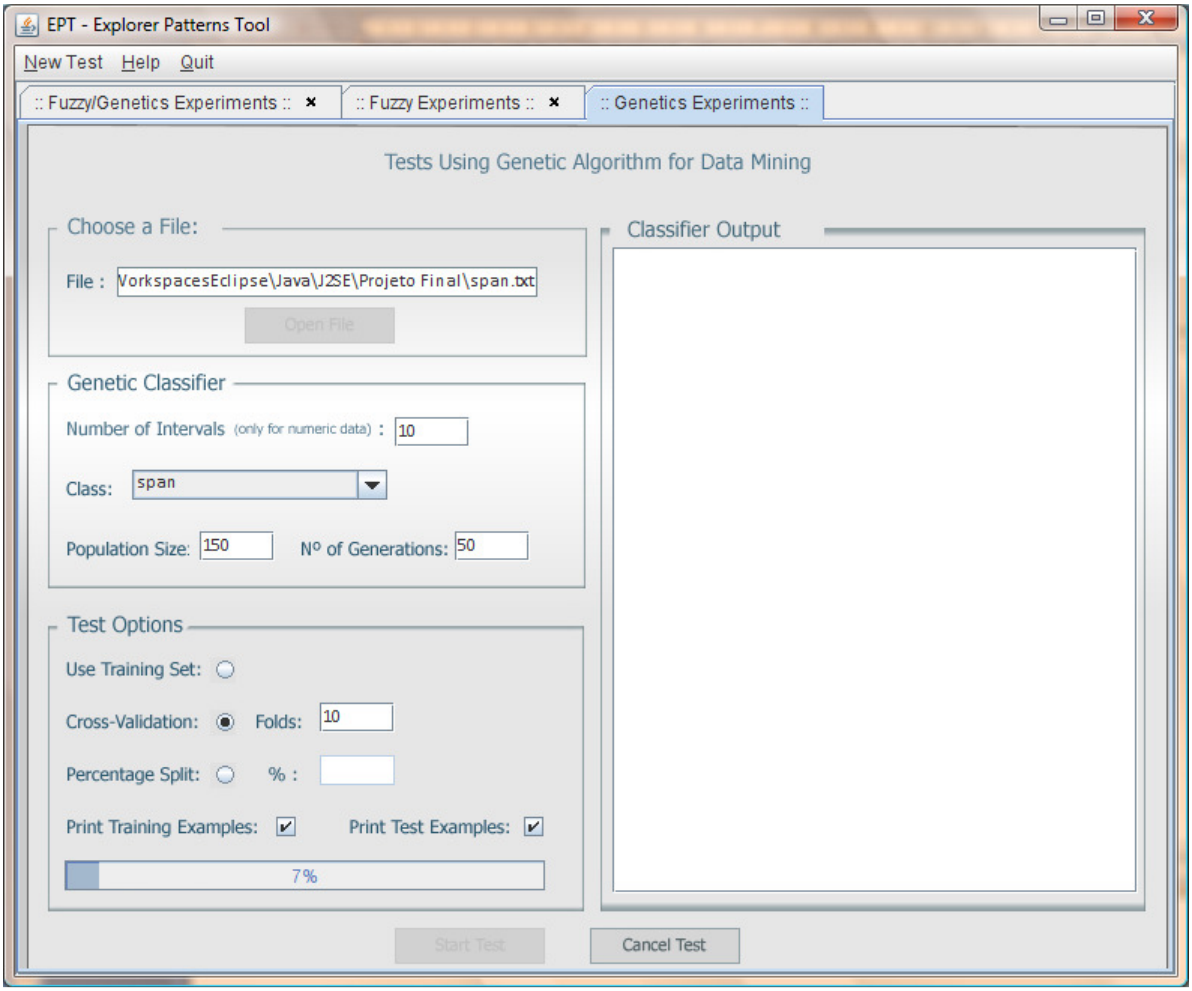

Figura 5.7: Execução de experimento utilizando o algoritmo genético implementado.

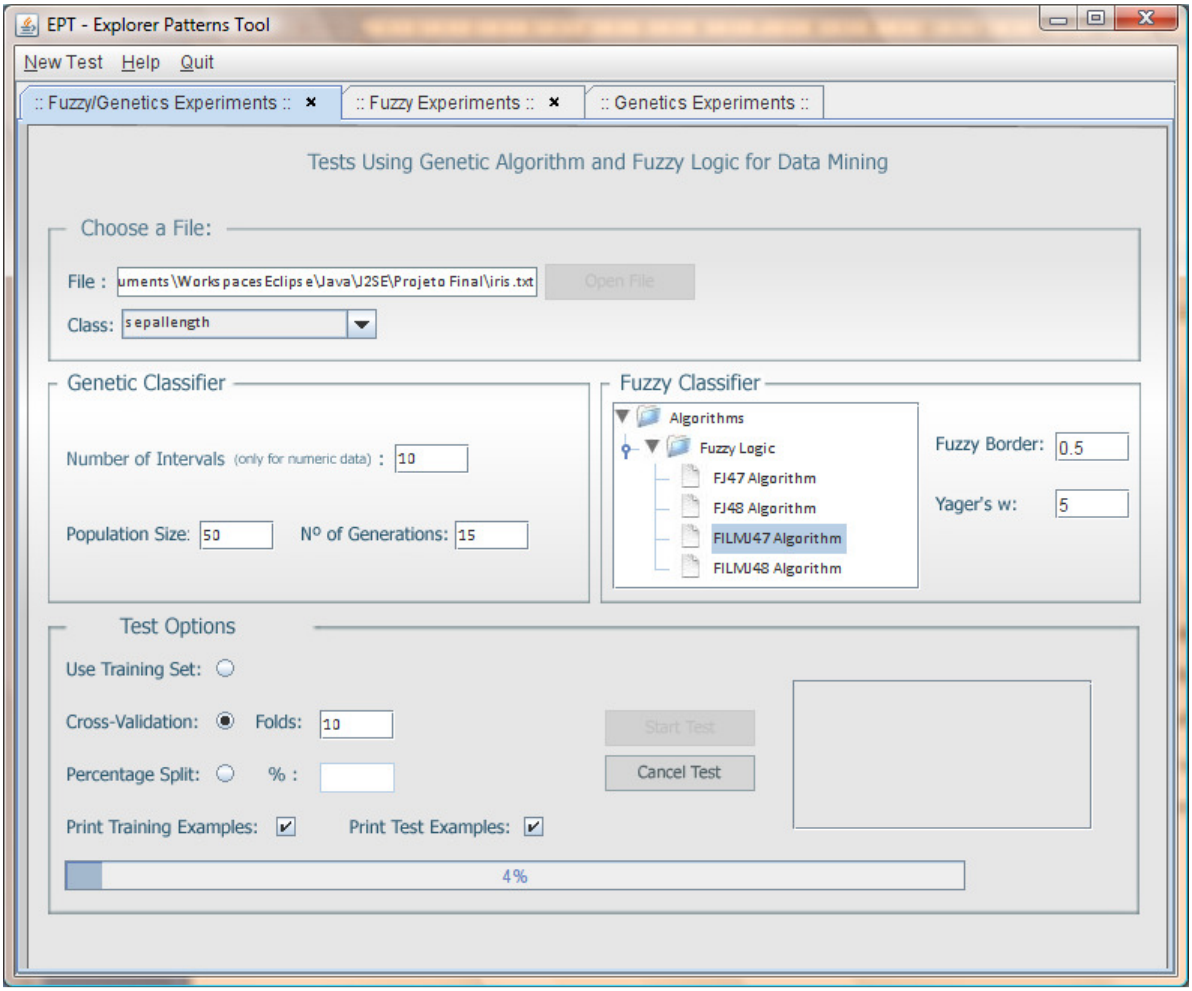

Figura 5.8: Execução de experimento utilizando o algoritmo genético implementado e comparando o resultado com a árvore *fuzzy* escolhida.

5.3.1 Relatório de Resultados

O *Explorer Patterns Tool* gera um relatório, que pode ser salvo em disco, e que exibe as informações acerca do teste feito pelo usuário. Neste teste, o usuário pode solicitar ainda que sejam impressos os exemplos utilizados para treinamento e validação para o algoritmo escolhido, caso sejam escolhidos os métodos de experimentação de validação cruzada ou percentagem de treino, que serão mais bem explanados em 6.2.2 e 6.2.3 respectivamente.

6 EXPERIMENTOS REALIZADOS

Para avaliar o desempenho do algoritmo genético implementado para este trabalho, foram realizados experimentos em algumas bases de dados, onde se avaliou a acurácia das regras encontradas. Foram feitos também experimentos para a comparação do desempenho do algoritmo implementado, baseado na computação evolutiva, com os algoritmos implementados para a ferramenta EFT, baseados em árvores nebulosas e clássicas, e usados para descoberta de padrões em bases de dados. Os experimentos foram feitos, utilizando o método de validação cruzada (*cross validation*) sobre quatro bases de dados. Este capítulo relata os experimentos realizados.

6.1 Bases de Dados Utilizadas

Os experimentos foram realizados sobre as mesmas bases de dados utilizadas em (SOUZA, 2008) e que também foram utilizadas para testar o EFT. As características de cada base de dados são descritas a seguir:

- Iris (FISHER, 1988) – esta base de dados é utilizada para ilustrar análise discriminante (SOUZA, 2008). A base de dados Iris tem ao todo 150 exemplos, que descrevem três tipos de flores*: Iris-setosa, Iris-versicolor* e *iris-virginica.* Cada objeto desta base possui quatro atributos reais (numéricos) e um atributo nominal, o atributo *class* que classifica cada objeto como um dos três tipos de flores descrito acima. Cada tipo de flor é a classe de 50 objetos cada.

- *Segment-Challenge* (GROUP, 1990a) – esta base de dados foi obtida a partir da seleção aleatória de objetos de uma base de dados de sete imagens de *outdoors* segmentadas para que seja possível a classificar todos os pixels. O tamanho desta base de dados é de 1500 objetos, e cada objeto possui 20 atributos, sendo 19 atributos reais (numéricos) e mais um atributo nominal,

chamado de *class* e que possui o seguinte domínio: *brickface, sky, foliage, cement, window, path, e grass.*

- *Segment-Test* (GROUP, 1990a) – esta base de dados é um subconjunto da base de dados *Segment-Challenge* com 810 objetos.

- *SPAMBASE* (HOPKINS et al., 2001) – base codificada de *e-mails* construída por pesquisadores da HP (SOUZA, 2008). Possui 4601 *e-mails* pessoais cedidos pelo pesquisador George Forman, codificados em 58 atributos – 57 atributos de domínio real (numéricos) e um atributo binário, o atributo *span* (o valor 1 denota que o objeto não é um *span* o valor 0 denota que este objeto trata-se de um *span*).

A tabela 6.1 faz um breve comparativo entre as bases de dados supracitadas.

Tabela 6.1: Informações sobre os conjuntos de dados (SOUZA, 2008)

Conjuntos de Dados	Domínio	Classes	Atributos de entrada	Qtde. de instâncias
SPAMBASE	Comercial	2	57	4601
Segment-challenge	Comercial	7	19	1500
Segment-test	Comercial	7	19	810
Iris	Biológico	3	4	150

6.2 Métodos de Experimentação

As subseções seguintes descrevem os três métodos de experimentação disponíveis na ferramenta EPT: conjunto de treino (*training set*), validação cruzada (*cross-validation*) e porcentagem de treino (*percentage split*).

6.2.1 Conjunto de Treino (*Training Set*)

Este método de experimentação utiliza todos os objetos da base de dados para treinar (conjunto de treino) e testar (conjunto de validação) o algoritmo utilizado (SOUZA, 2008 apud EGGERMONT, 2005).

6.2.2 Validação Cruzada (*Cross-validation*)

Neste método a base de dados *B* é aleatoriamente dividida em *k* subconjuntos mutuamente exclusivos de tamanhos aproximadamente iguais chamados *folds* (B_1, B_2, ..., B_k) (KOHAVI, 1995). O algoritmo é executado *k* vezes; cada vez $t \in \{1, 2, ..., k\}$ o algoritmo é treinado utilizando $B \setminus B_t$ e testado em B_t (KOHAVI, 1995).

O procedimento acima é repetido até que os *k* subconjuntos sejam utilizados como conjunto de teste (SOUZA, 2008). A acurácia final é obtida através do cálculo da acurácia média dos *folds* (SOUZA, 2008 apud SILVA, 2007).

6.2.3 Percentagem de Treino (*Percentage Split*)

Neste método uma porcentagem *p* dos exemplos de uma base de dados é utilizada como conjunto de treino. Essa porcentagem *p* é definida pelo usuário e os objetos do conjunto de treino são escolhidos aleatoriamente. As instâncias que não foram selecionadas para o conjunto de teste, são então utilizadas no conjunto de treinamento (SOUZA, 2008 apud GONCALVES, 2007).

6.3 Procedimentos Experimentais

Os experimentos foram realizados sobre o algoritmo genético implementado para este trabalho e sobre os algoritmos implementados para a ferramenta EFT: J47, J48, FILMJ47, FILMJ48, FJ47 e FJ48. O primeiro experimento visa comparar os resultados obtidos pelo AG deste trabalho com os valores obtidos nos testes realizados em SOUZA, 2008 – comparações entre as árvores clássicas e nebulosas. No segundo experimento, o algoritmo genético foi executado utilizando diferentes combinações de tamanho de população, número de gerações executadas e tamanho de intervalos utilizados para classificação.

O objetivo do primeiro experimento é realizar uma comparação entre os índices de acurácia obtidos pelo algoritmo implementado com os índices obtidos pelos algoritmos baseados em árvores clássicas e nebulosas que foram implementados no EFT. O primeiro experimento foi feito sobre todas as bases de dados mostradas anteriormente. Para este experimento foram utilizadas duas modalidades de teste: divisão por porcentagem (*percentage split*) e validação cruzada (*cross validation*). Para a divisão por porcentagem, foram feitas 300 execuções do algoritmo implementado, nas quais foram utilizados 60% do

conjunto de dados respectivo para treinar o algoritmo e os outros 40% restantes foram utilizados para validação das regras encontradas. Para a validação cruzada, as bases de dados foram divididas em 10 *folds*.

O objetivo do segundo experimento é realizar um estudo da relação entre o número de gerações implementadas e as taxas de acurácia resultantes, executando este algoritmo 300 vezes utilizando a validação cruzada, dividindo a base de dados *Spambase* em 10 folds, obtendo e comparando os resultados.

6.4 Resultados e Discussões

PRIMEIRO EXPERIMENTO – COMPARAÇÃO ENTRE ALGORITMOS

A tabela 6.2 exibe a média aritmética obtida para os resultados obtidos nos testes com o algoritmo genético, onde a coluna AG mostra os resultados obtidos pelo algoritmo genético implementado: V.C. representa os resultados obtidos para os testes que utilizaram a validação cruzada e D.P. representa os resultados obtidos pelos testes utilizando o método de divisão por porcentagem. As colunas seguintes representam os resultados obtidos em (SOUZA,2008).

Tabela 6.2: Resultados das Comparações entre as Taxas de Acurácia Obtidas pelo Algoritmo Genético Implementado e as Árvores *Fuzzy* Implementadas no EFT

Conjuntos de Dados	AG		FILMJ47	FILMJ48	FJ47	FJ48
	V.C.	D.P.				
SPAMBASE	0,520	0,622	0,853	0,857	0,614	0,645
Segment-challenge	0,820	0,832	0,874	0,874	0,838	0,841
Segment-test	0,830	0,845	0,847	0,835	0,790	0,798
Iris	0,726	0,784	0,869	0,907	0,803	0,837

O primeiro experimento mostrou taxas de acurácia maiores que 60% em todas as bases para o método de divisão por porcentagem, porém apenas para a base de dados

Segment-test a taxa de acurácia foi superior às taxas obtidas pelos outros algoritmos. Este desempenho inferior é explicado pelo caráter heurístico dos algoritmos genéticos, além da precisão obtida pelos algoritmos de árvore clássicas e *fuzzy* ao trabalharem com atributos contínuos e que não é possível no algoritmo implementado já que este algoritmo utilizou discretizações simples para trabalhar com estes valores. Os valores de acurácia também poderiam ser modificados caso outros operadores fossem utilizados neste algoritmo ou caso fosse utilizado um método de discretização mais preciso.

SEGUNDO EXPERIMENTO – COMPARAÇÃO ENTRE GERAÇÕES

A figura 6.1 exibe os resultados obtidos para o segundo experimento. Este experimento mostrou que o índice de acurácia obtido pelo algoritmo implementado é aumentado à medida que o número de gerações também é aumentado.

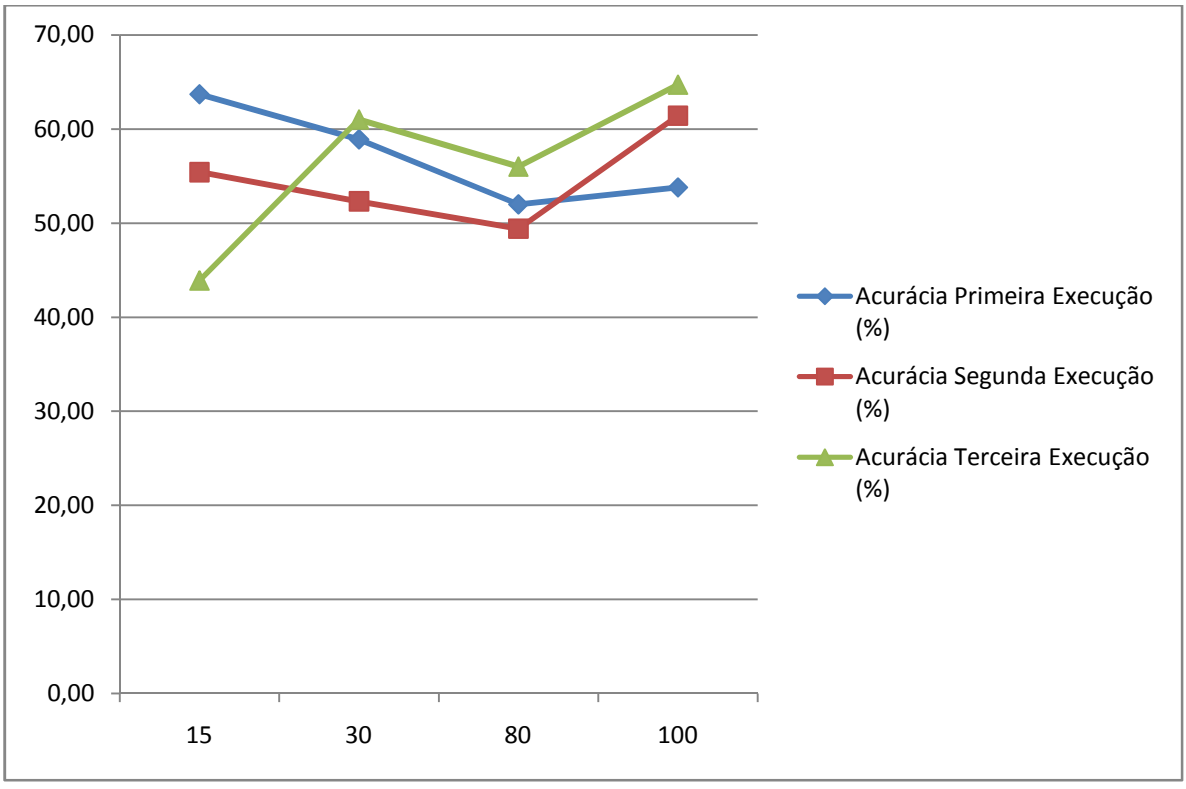

Figura 6.1: Acurácia X número de gerações

7 CONCLUSÃO

O domínio das informações contidas em um banco de dados, através do real conhecimento de seu conteúdo e aplicação é um diferencial em várias áreas. Empresas privadas podem utilizar técnicas de extração de dados para melhorar sua competitividade no mercado, ou ainda governos podem utilizar a mineração de dados para aplicar seus recursos de forma mais efetiva. Sendo assim, pode-se concluir que o processo de KDD, destacando a etapa de Mineração de Dados tem grande importância e aplicação prática.

Um classificador genético baseado em regras de predição pode ser construído a partir de um conjunto de exemplos de treinamento e testado em um conjunto de exemplos de testes, sendo estes exemplos representantes do ambiente que se deseja conhecer e das situações que se deseja predizer futuramente.

Este trabalho implementou um algoritmo genético utilizando técnicas que, segundo a literatura pesquisada, maximizavam o desempenho de um algoritmo genético para encontrar regras de predição. Todavia, o algoritmo implementado obteve acurácia superior a acurácia obtidas por técnicas de árvores clássicas apenas em uma base de dados. Este algoritmo obteve taxas de acurácia menores que todos os métodos de árvores *fuzzy*.

A comparação do uso de um algoritmo utilizando lógica nebulosa aplicado a mineração de dados com um algoritmo genético desenvolvido para a mesma tarefa não é um tema abordado freqüentemente, ao contrário da união destas duas técnicas para realizar a tarefa de mineração de dados, tema abordado em muitos trabalhos pesquisados.

Este trabalho propõe as seguintes possibilidades de trabalhos futuros:

- A união das técnicas de lógica nebulosa e genética para melhorar a acurácia das regras;

- A implementação de um algoritmo genético que proponha uma função de avaliação, um modo de seleção de indivíduos, um modo de mutação diferentes do utilizado pelo algoritmo implementado neste trabalho, além da não discretização dos valores contínuos – o tratamento a estes valores seria dado por meio da interpolação linear;

- A Implementação de métodos de seleção de atributos para diminuir o tempo de execução dos algoritmos implementados.

- Uso de redes neurais auto organizáveis para descoberta de regras de predição.

REFERÊNCIAS BIBLIOGRÁFICAS

CARVALHO, Deborah R. *Árvore de Decisão / Algoritmo Genético para Tratar o Problema de Pequenos Disjuntos em Classificação de Dados.* 2005. 173 f. Tese (Doutorado em Ciências em Engenharia Civil) – Programas de Pós-Graduação em Computação de Alto Desempenho/Sistemas Computacionais e Programa de Engenharia Civil da Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2005.

CARVALHO, Deborah R., FREITAS, Alex A. *Uma revisão de métodos de Niching para algoritmos genéticos.* Tuiuti: Ciência e Cultura, n. 29, FACET 04, pp. 97-118, Curitiba, 2002.

COX, Earl. *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration.* Elsevier/Morgan Kaufmann, 2005.

DOHERTY, C. Gregory. *Fundamental Analysis Using Genetic Programming For Classification Rule Induction.* California, 2001.

FAYYAD, Usama, SHAPIRO-PIATETSKY, Gregory, SMITH, Padhraic. *From Data Mining to Knowledge Discovery in Databases.* 1996.

FERNANDES, Anita Mª da Rocha, RAMPELOTTI, Flávio M. *Data Mining*. In: *Inteligência Artificial – noções gerais.* Visual Book, 2003. p. 129 – 145.

FERNANDES, Anita Mª da Rocha, COSTA Jr., Ilaim. *Algoritmos Genéticos*. In: *Inteligência Artificial – noções gerais.* Visual Book, 2003. p. 116 – 127.

FISHER, M. M. R. Iris. 1988. Último acesso em 20 de dezembro de 2008. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>.

CASTRO, Leandro de, ZUBEN, Fernando Von. *Algoritmos Genéticos (AG's)*. Último acesso em 20 de dezembro de 2008. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_02/topico9_02.pdf>

FREITAS, Alex A. *A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery*. Curitiba, 2001.

FREITAS, Alex A. *A Review of Evolutionary Algorithms for Data Mining.* Canterbury, 2007.

FREITAS, Alex A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, 2002

FUCHS, Gabriel. *Data Mining- If Only Really Were about Beer and Diapers,* Jan. 2004. Último acesso em 01 de Novembro de 2008.
Disponível em: <http://www.dmreview.com/news/1006133-1.html>

GIORDANA, Attilio, NERI, Filippo. *Search-Intensive Concept Induction*. In*: Evolutionary Computation.* MIT Press, 1995. P. 375 – 419.

GOEBEL, Michael, GRUENWALD Le. *A Survey of Data Mining and Knowledge Discovery Software Tools.* ACM SIGKDD Explorations, 1999.

LINDEN, Ricardo. *Algoritmos Genéticos. Uma importante ferramenta da Inteligência Computacional.* Brasport, 2006.

KOHAVI, R. *A study of cross-validation and bootstrap for accuracy estimation and model selection*. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, n. 2, p. 1137–1143, 1995.

MITCHELL, Melanie. *An Introduction to Genetic Algorithms.* Cambridge, MA. MIT Press, 1996

PAPPA, Gisele L. *Seleção de Atributos Utilizando Algoritmos Genéticos Multiobjetivos.* 2002. 85 f. Dissertação (Mestrado em Informática Aplicada) - Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná, Curitiba, 2002.

PAPPA, Gisele L., FREITAS, Alex A. *Towards a Genetic Programming Algorithm for Automatically Evolving Rule Induction Algorithms.* Canterbury, 2004.

RAS, Zbigniew W., ZEMANKOVA, Maria. *Methodologies for Intelligent Systems: 8th International Symposium, ISMIS '94, Charlotte, North Carolina, USA, October 16-19, 1994 : Proceedings*. Springer, 1994.

SOUZA, Erick Nilsen Pereira de. *Explorer Fuzzy Tree: uma ferramenta para experimentação de técnicas de classificação baseadas em árvores de decisão fuzzy*. *In ERBASE 2008 – WTICG-BASE (Workshop de Trabalhos de Iniciação Científica e Graduação Bahia, Alagoas e Sergipe)* Salvador, Bahia, 2008.

WU, Yi-Ta, AN Yoo Jung, GELLER, James, WU, Yih-Tyng. *A Data Mining Based Genetic Algorithm. In Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Second International Workshop on Collaborative Computing, Integration and Assurance.* SEUS-WCCIA, 2006.

APÊNDICE A – MANUAL PARA UTILIZAÇÃO DO *EXPLORER PATTERNS TOOL*

O *Explorer Patterns Tool* oferece uma interface gráfica para facilitar a realização dos experimentos sobre os algoritmos implementados para a ferramenta.

Cada uma das telas da ferramenta possui o botão *Open File* que permite que seja carregado o arquivo que contém os exemplos da base de dados que serão utilizados para testar e treinar os algoritmos. Este arquivo deve estar no formato descrito ilustrado pela figura A.1.

Caso o usuário selecione o botão *Start Test* de uma das telas da ferramenta, sem ter carregado um arquivo de dados a mensagem de erro ilustrado em A.2 é exibida e o teste não é iniciado. Cada tela da ferramenta EPT é descrita nas subseções a seguir.

```
sepallength
sepalwidth
petallength
petalwidth
class Iris-setosa Iris-versicolor Iris-virginica

@data
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa
5.2,3.5,1.5,0.2,Iris-setosa
5.2,3.4,1.4,0.2,Iris-setosa
4.7,3.2,1.6,0.2,Iris-setosa
```

Figura A.1: Formato de arquivo de dados a ser lido pelo *Explorer Patterns Tool* (SOUZA, 2008)

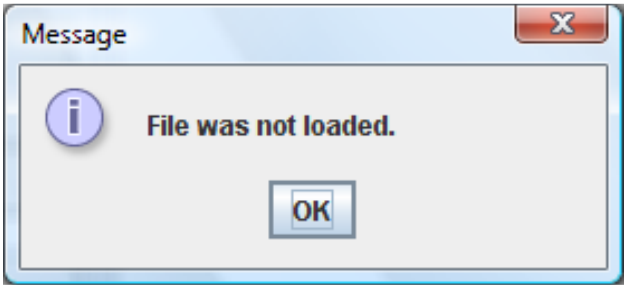

Figura A.2: Mensagem de erro gerada caso um arquivo não tenha sido carregado.

A.1 Tela Principal

A figura A.3 mostra a tela principal do *Explorer Patterns Tool* e seu menu, que permite que seja selecionada uma modalidade para a realização dos experimentos. As opções do menu são descritas a seguir.

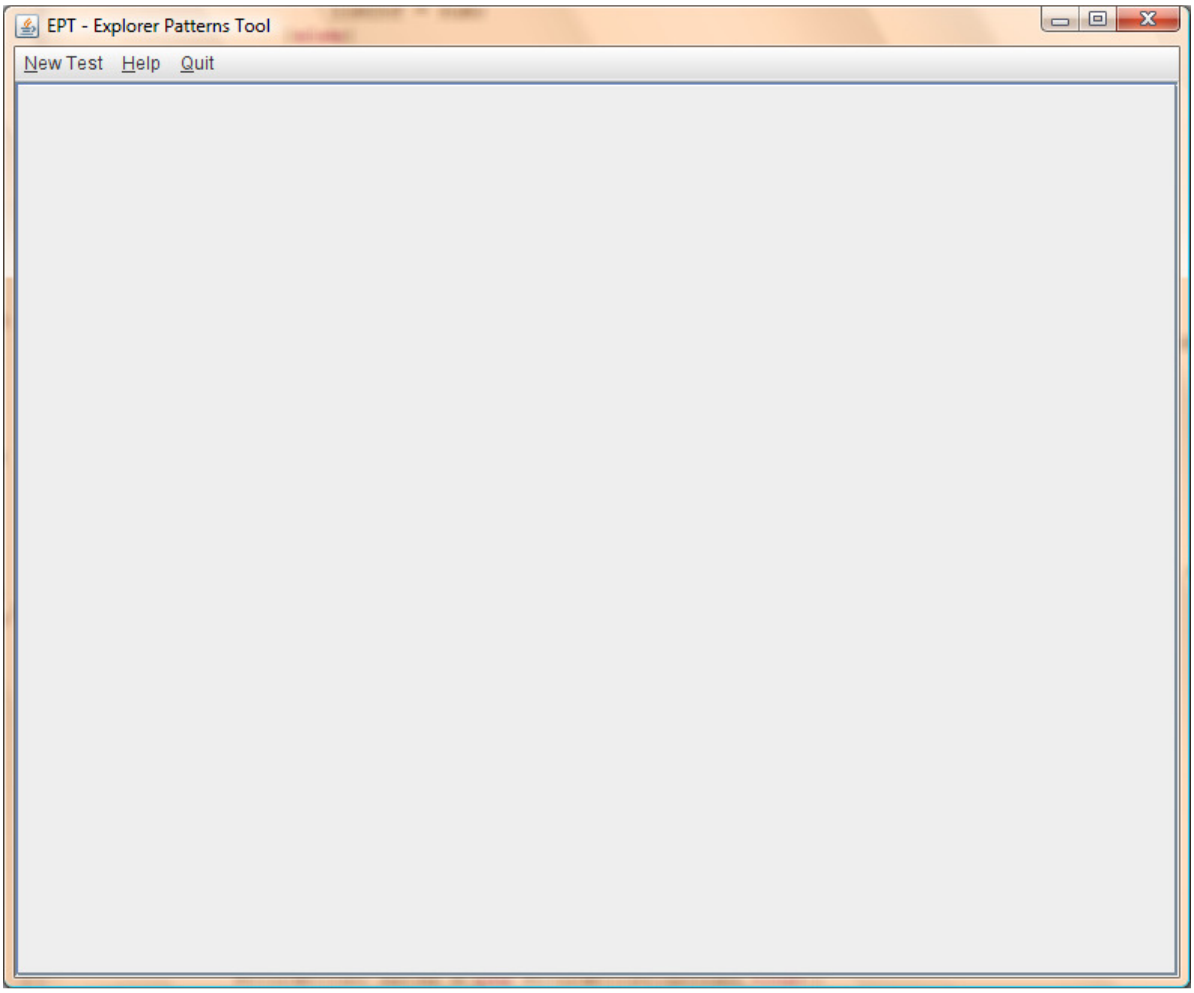

Figura A.3: Tela principal do EPT

O menu *New Test* oferece as seguintes opções:

- *Fuzzy Algorithms* (Teclas de atalho: Alt + E) – se escolhida esta opção uma nova janela é aberta. Nesta nova janela é possível a realização de experimentos utilizando os algoritmos baseados na lógica nebulosa para indução de regras;

- *Genetic Algorithms* (Teclas de atalho: Alt +N) – se escolhida esta opção uma nova janela é aberta. Nesta nova janela é possível a realização de experimentos utilizando um algoritmo evolucionário para descoberta de regras de classificação;

- *Compare Genetics and Fuzzy* (Teclas de atalho: Alt + C) - se escolhida esta opção uma nova janela é aberta. Nesta nova janela é possível realizar um experimento onde as abordagens genética e nebulosa são comparadas;

O menu Help carrega a ajuda da ferramenta. O menu *Quit* (Teclas de atalho: Alt + Q) finaliza o programa.

A.2 Tela para Experimentos Utilizando Lógica Nebulosa

Acessando a opção ***Fuzzy Algorithms*** (teclas de atalho Alt + E), através do menu localizado na tela principal, é aberta uma nova janela para que sejam realizados experimentos baseados em árvores de decisão nebulosas e clássicas. Esta janela possui as mesmas funcionalidades oferecidas pelo EFT além de novas funcionalidades. A figura A.4 ilustra esta tela.

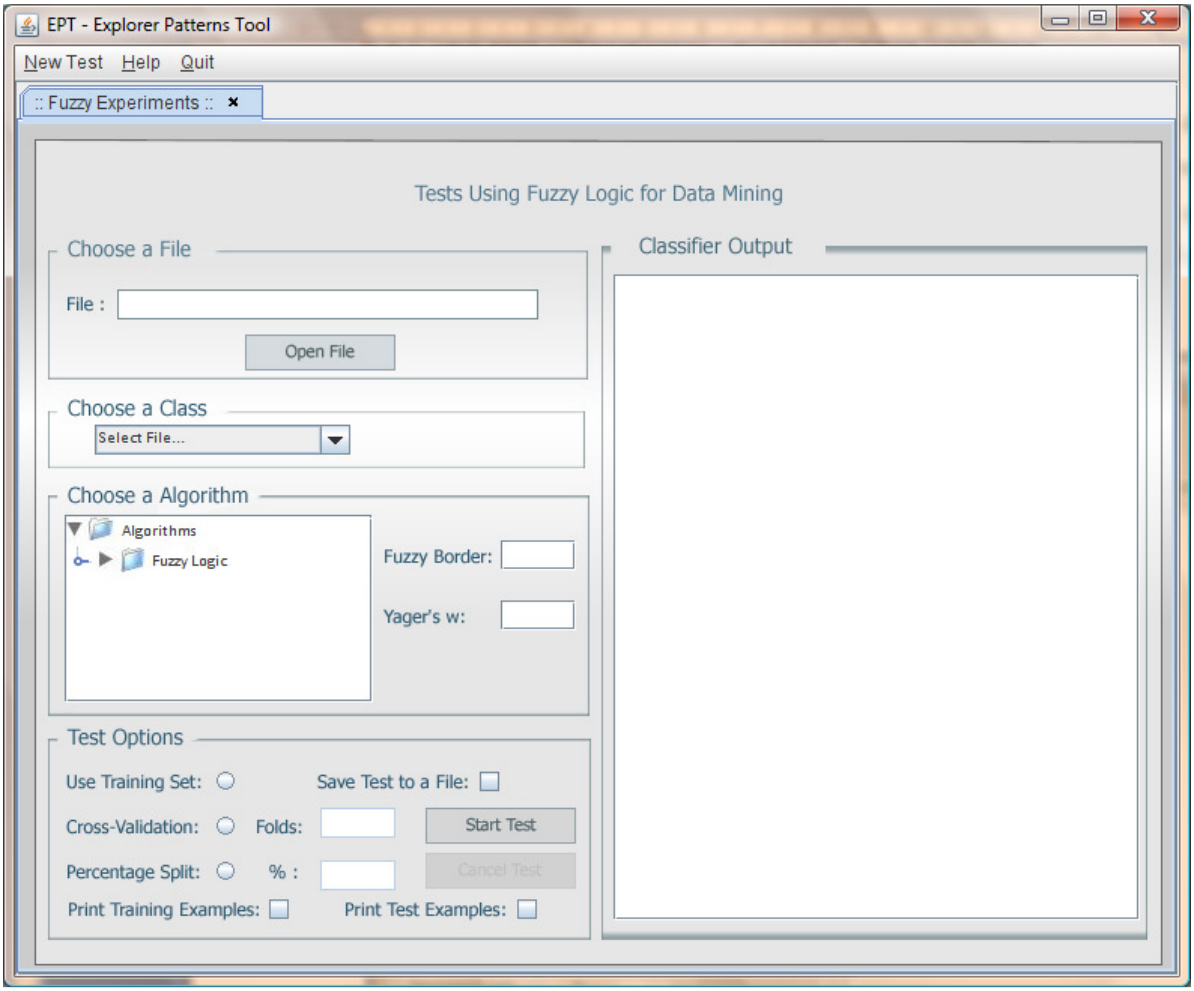

Figura A.4: Tela para experimentos utilizando árvores de decisão nebulosas

Após a escolha do arquivo contendo a base de dados, o rótulo de cada atributo dessa base de dados é mostrado na combo localizada na opção *Choose a class*. O atributo escolhido através dessa combo será o atributo objetivo considerado pelo algoritmo escolhido.

Em *Choose a Algorithm* permite que o usuário escolha um dos algoritmos nebulosos implementados. Os algoritmos disponíveis são os mesmos algoritmos implementados no EFT. Para cada algoritmo, devem ser informados parâmetros de entrada (*Fuzzy Border* e/ou *Yager's Border*), que variam de acordo com o algoritmo solicitado. A ferramenta habilita para preenchimento apenas os parâmetros realmente utilizados pelo algoritmo escolhido pelo usuário.

Em *Test Options* é possível escolher o tipo de teste a ser realizado, como é possível salvar os resultados dos experimentos em um arquivo, como a escolha sobre a exibição ou não dos exemplos utilizados para treino e teste do algoritmo.

Em *Classifier Output* é mostrado os resultados dos testes realizados – a árvore de classificação e os resultados de acurácia para as árvores clássicas e as árvores *fuzzy*.

O botão *Start Test* inicializa os testes apenas se todos os campos obrigatórios tenham sido devidamente preenchidos, e o botão *Cancel Test* cancela a realização deste teste em tempo de execução.

Caso o usuário tenha deixado de informar alguns dos parâmetros obrigatórios para a realização do experimento escolhido, o *Explorer Patterns Tool* exibe a mensagem ilustrada na figura A.5.

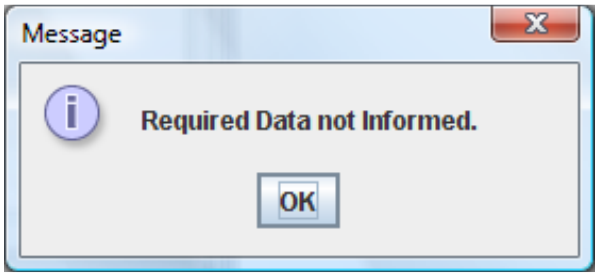

Figura A.5: Mensagem de erro gerada caso algum item requerido não seja informado.

Caso tenha sido escolhida a opção *Save Test to a File* após o término do teste é exibida uma janela ilustrada na figura A.6 para que o usuário possa escolher o local onde seu arquivo contendo os resultados do teste será criado.

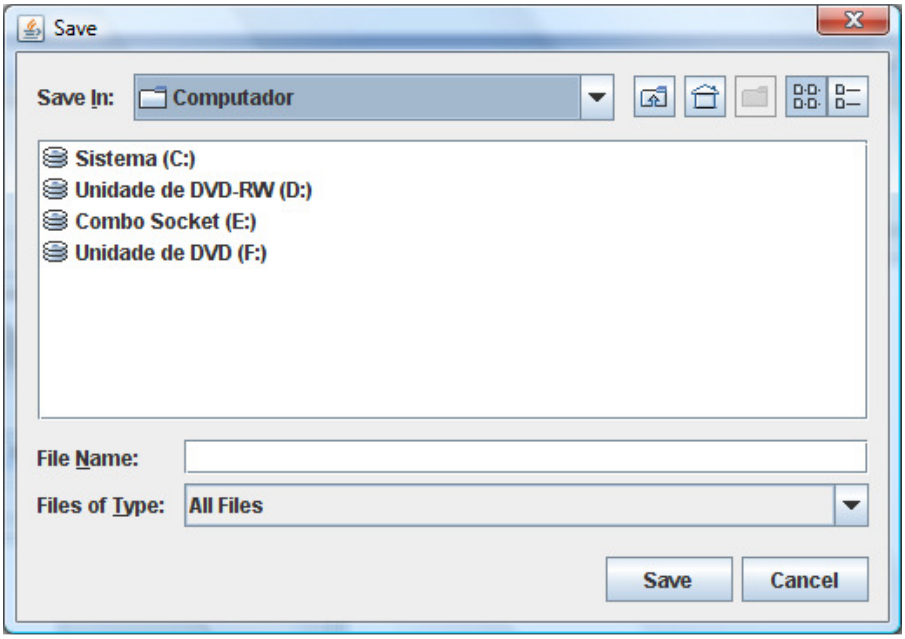

Figura A.6: Janela para gravação de arquivo de resultados para experimentos utilizando Lógica Nebulosa.

A.3 Tela para Experimentos Utilizando Algoritmos Genéticos

Acessando a opção ***Genetic Algorithms*** (teclas de atalho Alt + N), através do menu localizado na tela principal, é aberta uma nova janela para que sejam realizados experimentos baseados em um algoritmo genético. A figura A.7 ilustra esta tela.

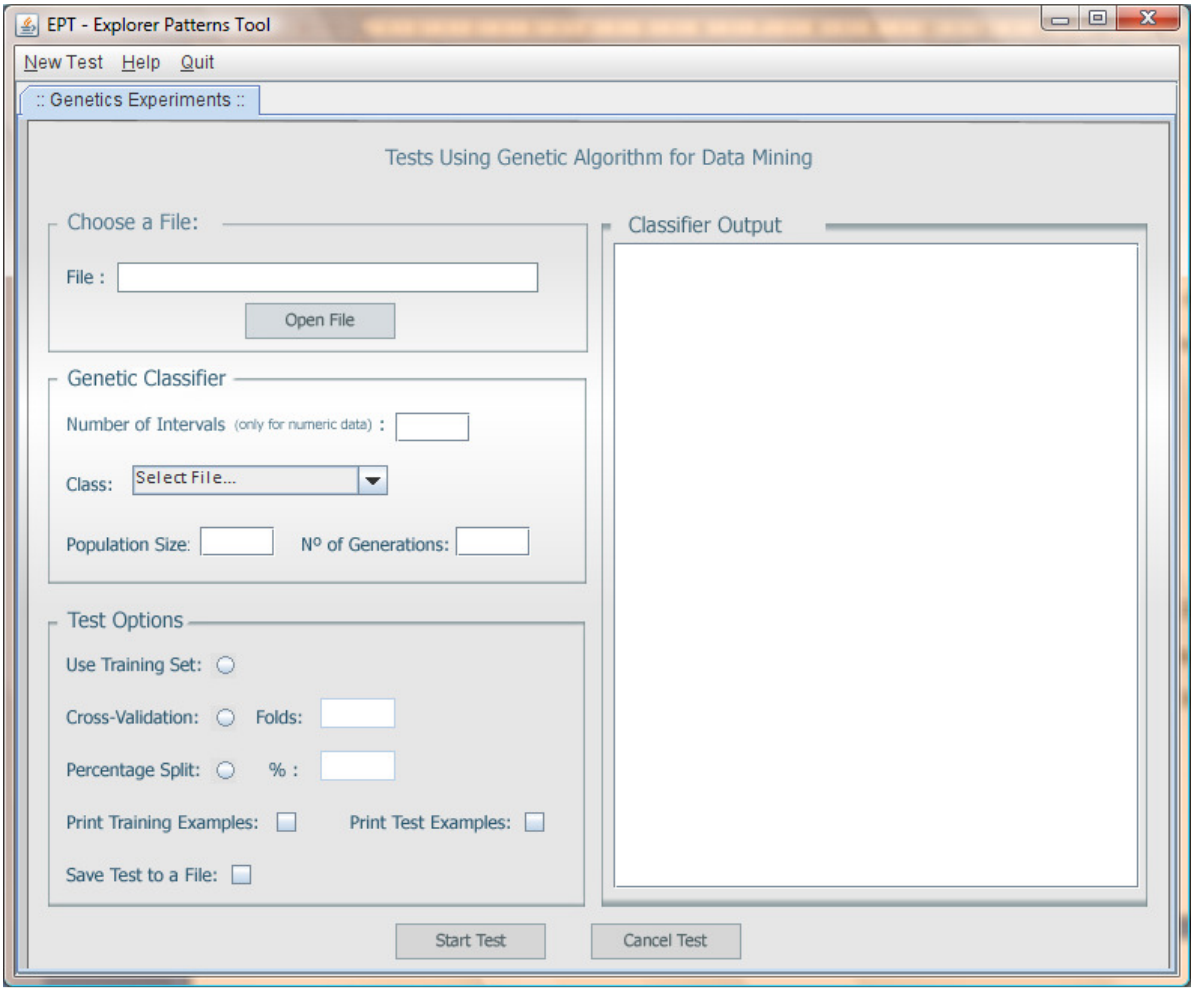

Figura A.7: Tela para experimentos utilizando um algoritmo evolucionário

Assim como descrito para a tela de experimentos com lógica *fuzzy*, o botão Open File permite que seja carregado o arquivo que contém os exemplos da base de dados que serão utilizados para testar e treinar os algoritmos e, caso o usuário selecione o botão *Start Test* sem ter carregado um arquivo de dados a mensagem de erro ilustrada em A.4 é exibida e o teste não é iniciado.

Em *Genetic Classifier*, devem ser informados os parâmetros necessários para a execução do AG. O campo *Number of Intervals* permite que seja informado pelo usuário o número de intervalos que será dividido os atributos contínuos quando estes forem discretizados como descrito na seção 5.2. A combo *Class* define qual será o atributo objetivo considerado pelo algoritmo genético, dentre os atributos da base de dados escolhida. Os campos *Population Size* e *Nº of Generations* permitem que o usuário informe o tamanho da

população do algoritmo genético (quantidade de indivíduos) e o número de execuções que serão realizadas para o AG (número de gerações), respectivamente.

Em *Test Options* é possível escolher o tipo de teste a ser realizado, como é possível salvar os resultados dos experimentos em um arquivo, como a escolha sobre a exibição ou não dos exemplos utilizados para treino e teste do algoritmo.

Em *Classifier Output* é mostrado os resultados dos testes realizados – as regras de classificação.

O botão *Start Test* inicializa os testes apenas se todos os campos obrigatórios tenham sido devidamente preenchidos, e o botão *Cancel Test* cancela a realização deste teste em tempo de execução. Caso algum campo requerido não seja informado, a mensagem ilustrada na figura A.5 é mostrada ao usuário e o teste não é iniciado.

Caso tenha sido escolhida a opção *Save Test to a File* após o término do teste é exibida uma janela ilustrada na figura A.8 para que o usuário possa escolher o local onde seu arquivo contendo os resultados do teste será criado.

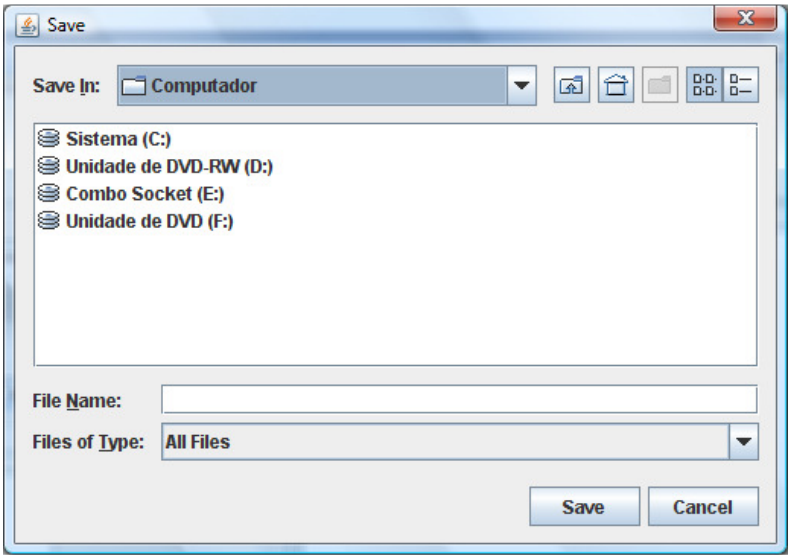

Figura A.8: Janela para gravação de arquivo de resultados para experimentos utilizando AG.

A.4 Tela para Experimentos Utilizando Algoritmos Genéticos e Lógica Nebulosa

Acessando a opção ***Compare Genetic and Fuzzy*** (teclas de atalho Alt + C), através do menu localizado na tela principal, é aberta uma nova janela para que sejam realizados experimentos que irão comparar o desempenho dos algoritmos utilizando lógica *fuzzy* oferecidos pelo EFT e o algoritmo genético implementado para este trabalho. A figura A.9 ilustra esta tela.

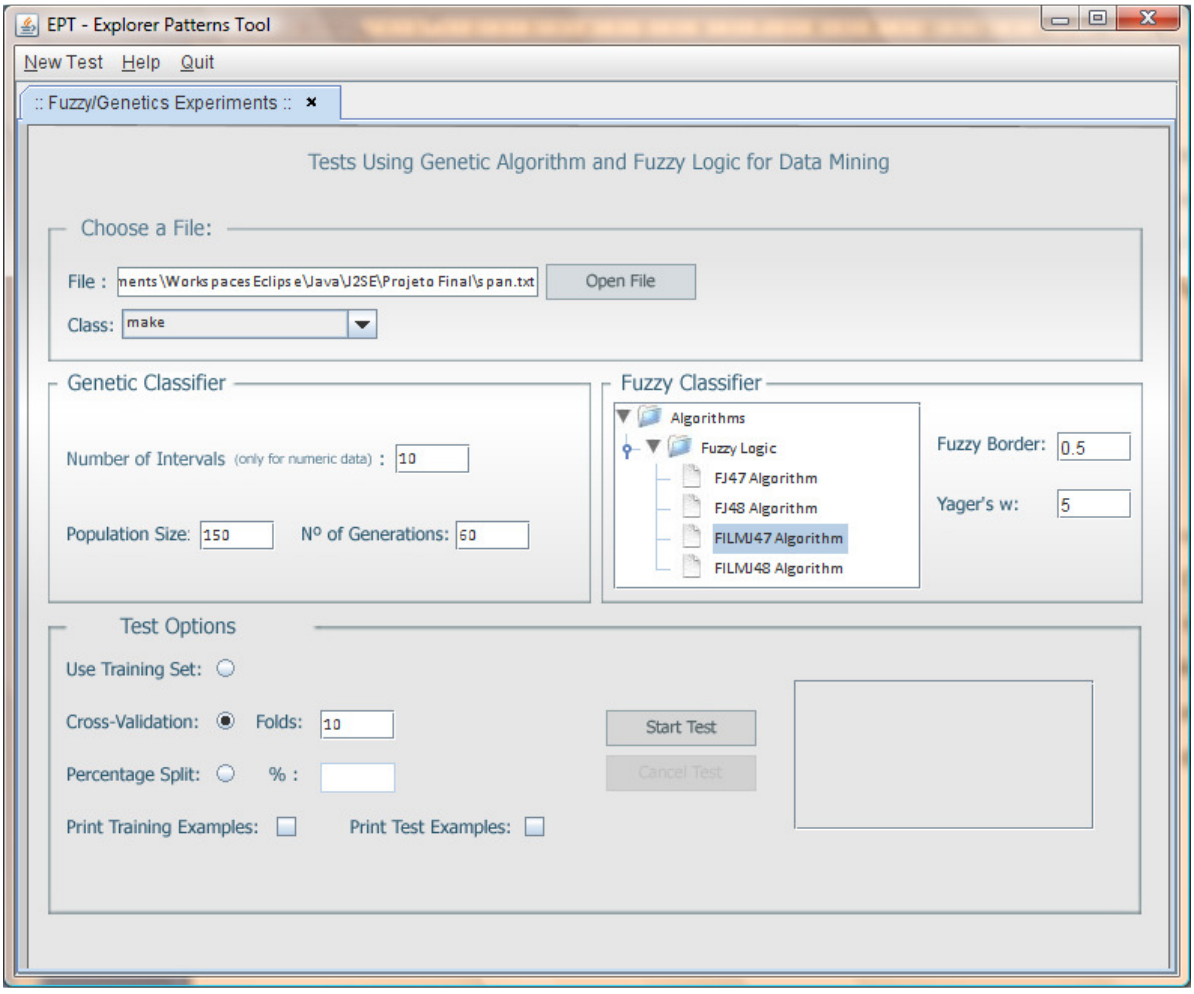

Figura A.9: Janela para realização de experimentos comparativos entre algoritmos nebulosos e genéticos

Esta tela foi desenvolvida para proporcionar um ambiente de comparação acerca do desempenho entre os algoritmos nebulosos e clássicos, implementados para o EFT e o algoritmo genético implementado para este trabalho, onde fosse garantido que os testes seriam realizados sobre os mesmos exemplos de treinamento e teste. O resultado para a comparação das técnicas é sempre salvo em um arquivo, pois o relatório gerado para esta tela possui um grande número de informações. O usuário pode escolher o local de gravação do arquivo, através da janela ilustrada pela figura A.10.

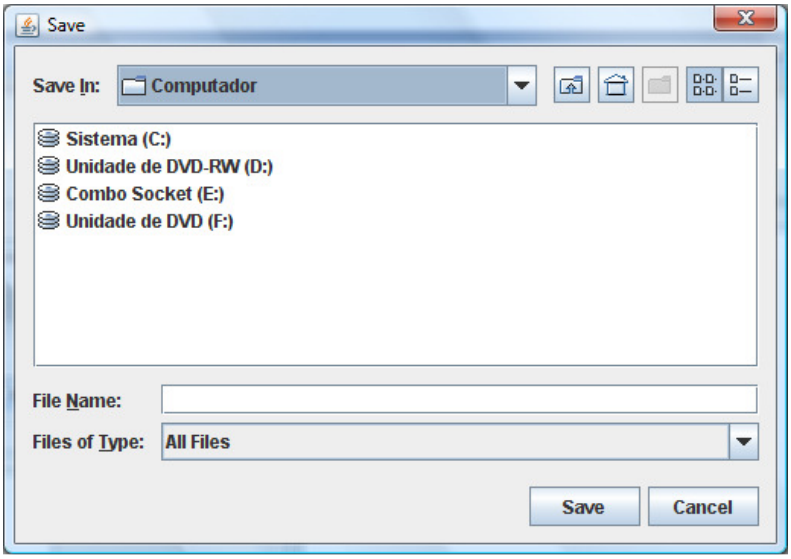

Figura A.10: Janela para gravação de arquivo de resultados dos testes comparativos.

Existe um painel localizado ao lado dos botões *Start Test* e *Cancel Test* além de uma barra de progresso para que o usuário possa acompanhar o estado do algoritmo.