FÁBIO MENDONÇA LOPES

ALGORITMO GENÉTICO RESTRITO POR LISTAS TABU NO CONTEXTO DE MINERAÇÃO DE DADOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática, Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná

Orientadora: Prof.ª Dr.ª Aurora T. R. Pozo

CURITIBA
2001

UFPR
Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática do aluno ***Fábio Mendonça Lopes***, avaliamos o trabalho intitulado "***Algoritmo Genético Restrito por Listas Tabu no Contexto da Mineração de Dados***", cuja defesa foi realizada no dia 30 de março de 2001. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 30 de março de 2001.

Profª. Dra. Aurora Trinidad Ramirez Pozo
Presidente - Orientadora

Profª. Dra. Maria Teresinha Arns Steiner
Membro Externo - MAT/UFPR

Profª. Dra. Silvia Regina Vergilio
DINF/UFPR

"Se você não tem oportunidade de fazer grandes coisas,
pode fazer pequenas coisas de uma forma grandiosa."

Brian Weiss

À minha amada esposa **Katia**, por estar sempre ao meu lado, em todos os momentos, me apoiando e incentivando, me motivando a ir sempre além,

DEDICO

Aos meus pais, **Maria** *(in memorian)* e **Irineu**, por todo o amor e carinho que recebi e continuo a receber,

OFEREÇO

AGRADECIMENTOS

O autor consigna sinceros agradecimentos:

À ***Professora Doutora Aurora Trindade Ramirez Pozo***, pela orientação ao longo desta Dissertação, pelas valiosas discussões conceituais, pelo incentivo e confiança que sempre me dispensou durante este trabalho e, principalmente, pela amizade conquistada, que muito enriqueceram minha formação pessoal e profissional.

À ***Professora Doutora Maria Terezinha Arns Steiner***, pela valiosa contribuição ao longo desta Dissertação, pelas palavras de incentivo, além da amizade e atenção sempre dispensadas durante esses anos.

À ***Professora Doutora Sílvia R. Vergílio***, pela prestimosa amizade e apoio ao longo deste Curso de Mestrado.

Ao ***Curso de Pós-Graduação em Informática***, em nome de todos os professores e funcionários, os quais contribuíram com minha formação acadêmica, transmitindo conhecimentos e fornecendo todo o suporte técnico, sem o qual a conclusão deste Curso não seria possível.

À ***Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)***, pela concessão da bolsa de Mestrado, a qual permitiu que este trabalho fosse realizado e concluído de maneira satisfatória.

À Bibliotecária ***Mariza Kampfert***, do Setor de Ciências Biológicas, pelo pronto atendimento sempre que solicitado, além do auxílio prestado na correção das referências bibliográficas.

Aos amigos ***Walter Antônio Pereira Boeger*** e ***Maria Regina Torres Boeger*** pela elaboração do Abstract e pela sincera amizade ao longo desses anos.

À minha amada esposa ***Katia Christina Zuffellato-Ribas***, pela sua ativa participação na minha vida pessoal e profissional, transmitindo sempre força e ânimo para que o futuro promissor seja uma mera consequência de um presente dedicado, por todo o seu amor e paciência ao longo deste trabalho, compreendendo todo meu nervosismo, ansiedades e falta de tempo. Eu te amo!

À minha querida cachorra ***Fiorella Zuffellato Lopes***, por sua adorável companhia nas infindáveis madrugadas na frente do computador, nas noites de chuva, sem jamais ter se enroscado nos fios e desligado a energia elétrica.

Aos meus pais, ***Maria Mendonça Lopes*** *(in memorian)* e ***Irineu Bernardo Lopes Martin***, por sempre me incentivarem e me apoiarem ao longo de minha vida.

À minha querida e amada sogra, ***Amelia Anita Zuffellato***, pelo apoio e incentivo manifestados durante os anos deste trabalho, além de todo o carinho de sempre.

À todos aqueles que, de uma forma ou de outra, colaboraram para a realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Resumo

O presente trabalho teve como objetivo a obtenção e implementação de um algoritmo de geração de um classificador, no contexto da Mineração de Dados. Este classificador utilizou Algoritmos Genéticos (AGs). Alguns dos fundamentos que justificaram a escolha deste paradigma, foram baseados em sua grande capacidade em lidar com ruídos, dados inválidos ou imprecisos e, sua facilidade de adaptação frente à diferentes domínios de dados. A principal contribuição do algoritmo projetado é a utilização de listas tabu restringindo o processo de seleção do AG. Esta restrição permite gerar um conjunto de regras potenciais para o classificador. Este tipo de estrategia foi proposta recentemente para trabalhar com funções multimodais e ainda não tinha sido avaliado seu comportamento no contexto de mineração de dados. Para análise da eficiência do algoritmo implementado foram realizados testes em cinco bases de dados, e os resultados comparados a 34 algoritmos classificadores. Posteriormente, foram realizados testes com adição de ruídos nas bases de dados. O algoritmo implementado demonstrou ser efeciente e robusto. A estratégia utilizada para manter a diversidade do AGs se mostrou válida, pois mesmo na utilização de populações menores, o algoritmo conseguiu manter sua precisão de classificação. A maior dificuldade encontrada no algoritmo foi o ajuste da medida de distância, parâmetro utilizado para as listas Tabu, o que afetou diretamente os resultados da precisão de classificação do algoritmo.

Abstract

The present work aims to obtain and to implement a algorithm of generation of a classifier, in the context of Data Mining. This classifier used Genetic Algorithms (AGs). The choice of this paradigm is partially justified on its great capacity in dealing with noise, invalid or inexact data, and its easy adaptation to different domains of data. The GA algorithm uses Tabu List to restrict the selection process. This restriction allows the creation a set of potential rules for the classifier tool. This strategy was proposed recently, for multimodal and multiobjective function optimization and this behavior had not still been evaluated in the context of Data Mining. For analysis of the efficiency of the algorithm, tests were performed on five databases and compared with 34 classifying algorithms. Later, tests with addition of noises to the databases were performed. The implemented algorithm was shown to be efficient and robust. The strategy used to keep the diversity in the searching process was considered valid, since even for smaller populations, the algorithm kept its accuracy of sorting. The biggest difficulty found in the algorithm was the adjustment of the measure of distance, parameter used for the Tabu lists, which directly affected the results of the accuracy of sorting of the algorithm.

1. Introdução

Nos últimos anos, a disponibilidade de recursos de gerar e armazenar dados vem aumentando significativamente. A utilização de códigos de barra para o controle da maioria dos produtos comerciais, a informatização de muitas operações de diferentes empresas ou simplesmente o avanço de ferramentas para a coleta e armazenagem de dados, têm gerado uma quantidade enorme de informações (CHEN; HAN; YU,1996).

Destes milhões de bases geradas, ricas em uma variedade de informações úteis, surge um novo interesse por parte das empresas, a descoberta do conhecimento em bases de dados (*knowledge discovery in databases - KDD*). Assim, a existência e a obtenção de informações úteis, com um nível de abstração maior e grande rapidez, tornou-se o novo foco de interesse empresarial (DEVLIN, 1997).

Aplicações de *KDD* com resultados positivos podem ser encontradas na área de doenças como previsão de recorrência de tumores, estimativa de sobrevida de pacientes, previsão meteorológica, predição de novos compostos químicos, dentre outras (NICOLETTI; SANTOS, 1996).

Várias técnicas foram criadas para automatizar a descoberta de informações, tendo como principal objetivo a obtenção rápida de informações para auxiliar empresas na tomada de decisões estratégicas, as quais chamamos de Mineração de Dados (DEVLIN, 1997).

Os Algoritmos Genéticos (AGs), um dos ramos da Computação Evolucionária, são inspirados na Teoria da Evolução de Charles Darwin. A idéia geral destes algoritmos é representar possíveis soluções de um problema como um "gene" de um indivíduo. Ao conjunto destes indivíduos é dado o nome de "população" que, imitando o processo natural de reprodução, possibilita a criação de um processo artificial de evolução da mesma, visando a busca de melhores soluções que representem o problema. Assim, a dinâmica de melhoramento dos indivíduos é realizada através do uso de operadores de recombinação (*crossover*) e "mutação", mediante uma medida de qualidade dos indivíduos (função de avaliação ou *fitness*), são selecionados os indivíduos mais bem adaptados para continuar no

processo de evolução das próximas gerações. Esta população é continuamente melhorada, até que algum critério de parada seja atingido (MITCHELL, 1997).

Quando comparamos estes algoritmos frente a outros no processo de descoberta de informações, os mesmos se destacam por certas vantagens, tais como: grande capacidade de lidar com dados inválidos ou imprecisos (FREITAS e LAVINGTON, 1998), a tratabilidade em termos de custo computacional (LANGLEY, 1996), o ajuste fino de parâmetros de acordo com o domínio (MITCHELL, 1997), e a possibilidade de paralelização e distribuição da carga de processamento (FREITAS; LAVINGTON, 1998).

Apesar das vantagens descritas, muitas pesquisas devem ser realizadas para explorar todo o potencial dos AGs. Um campo de interesse crescente refere-se a métodos de aprimoramento dos AGs adicionando-les capacidades de diversidade populacional, busca local e/ou distribuida. Entre estas estratégias podemos destacar a utilização de técnicas de criação de nichos (De JONG, 1975; GOLDBERG; RICHARDSON, 1987; DEB; GOLDBERG, 1989; MAHFOUD, 1992; HASSE, 2000) , e a integração de Algoritmos Genéticos a heurística busca Tabu (KURAHASHI; TERANO, 2000).

1.1. Objetivos

O presente trabalho teve por objetivo principal a implementação de um Algoritmo Genético restritos por listas Tabu para a tarefa de geração de classificadores.

Para alcançar as metas propostas, o presente trabalho foi dividido em objetivos específicos conforme segue:

- Estudo dos conceitos envolvidos na geração de um classificador através de Mineração de Dados, dentre eles:
  a. Técnicas de Mineração de Dados;
  b. Algoritmos Genéticos;
  c. Busca Tabu;
  d. Algoritmos Genéticos restrito por listas Tabu em Mineração de Dados.

- Implementação de um algorítmo para a geração de um classificador, baseada na filosofia exposta em d.

- Análise da eficiência da adoção da estratégia de integração adotada no âmbito de Mineração de Dados através da comparação do algoritmo implementado, frente a outras implementações que se utilizam de heurísticas tradicionais.

Foi utilizada a abordagem de integração entre Algoritmos Genéticos e busca Tabu. Esta abordagem foi proposta por KURAHASHI e TERANO (2000) para funções multimodais, porém ainda não foram realizados estudos no contexto de Mineração de dados.

1.2. Estrutura do Trabalho

Este trabalho foi organizado de forma a apresentar inicialmente, conceitos que envolvem o processo de Mineração de Dados e algumas de suas técnicas, os quais são vistos no Capítulo 2.

No Capítulo 3 é apresentada mais detalhadamente a estrutura de um Algoritmo Genético, bem como seu funcionamento e principais operadores. São apresentadas no Capítulo 4, algumas estratégias adotadas por outros pesquisadores, visando a uma melhora de desempenho no uso de Algoritmos Genéticos, como criação de nichos, e a combinação dos mesmos com busca Tabu (listas de restições).

Em seguida, no Capítulo 5, são descritas as etapas para a obtencão de um classificador e a heurística para a junção de regras, associada ao uso de um Algoritmo Genético, além do detalhamento da implementação do algoritmo proposto, discutindo os parâmetros adotados, bem como os detalhes do uso de listas de restrições no algoritmo.

Visando a analise do desempenho da implementação realizada, no Capítulo 6, é realizado um experimento através da utilização de algumas bases de dados, comparando-as frente a outros algoritmos (LIM; LOH; SHIH, 1999; HASSE, 2000). Finalmente, no Capítulo 7, são apresentadas algumas conclusões obtidas das análises realizadas no capítulo anterior, além de sugestões para trabalhos futuros.

2. Mineração de Dados

A descoberta do conhecimento em bases de dados tem se tornado um processo consideralvelmente interessante para pesquisadores de diferentes áreas (FRAWLEY; PIATETSKY-SHAPIRO; MATHEUS, 1991; SILBERSCHATZ; STONEBRAKER e ULLMAN, 1995; CHEN; HAN; YU, 1996; FAYYAD et al.,1996). Entre algumas das aplicações no processo de descoberta de conhecimento, podemos destacar:

- a descoberta de relações de associações e dependências entre dados.
- o agrupamento de um conjunto de dados selecionados a paritr de bases de dados (*data warehouse)*, podendo medir o grau de similaridade entre os grupos obtidos.
- construir um modelo que possa ser utilizado para categorizar ou classificar dados futuros.

Assim, neste capítulo, serão abordadas algumas definições sobre o processo de descoberta de conhecimento em bases de dados, bem como suas etapas. Dentre estas etapas, podemos citar a Mineração de Dados como aquela de maior interesse do presente trabalho, onde serão comentadas algumas das técnicas utilizadas para tal processo.

2.1. Processo de Descoberta de Conhecimento

Segundo FRAWLEY, PIATETSKY-SHAPIRO e MATHEUS (1991), o processo de descoberta do conhecimento em bases de dados, também conhecido como *knowledge discovery in databases* (*KDD*), significa a aplicação de um procedimento não trivial para identificação efetiva, coerente, potencialmente útil e prévia, de padrões desconhecidos em grandes bases de dados.

Devido a essas características incomuns, todo o processo *KDD* depende de uma nova geração de ferramentas e técnicas de análise de dados e envolve diversas etapas. A principal etapa, núcleo deste processo, chama-se Mineração de Dados, ou *Data Mining*, também conhecida como processo de arqueologia de dados, ou reconhecimento de padrões (CHEN; HAN; YU, 1996).

Até 1995, muitos autores consideravam os termos *KDD* e Mineração de Dados como sinônimos mas, em 1995, na Conferência Internacional de *KDD*, em Montreal, foi dada uma definição para cada um dos termos (ADRIAANS; ZANTINGE, 1996):

> " ...*KDD* será empregado para todo o processo de extração de conhecimento dos dados. Neste contexto, conhecimento significa relacionamento e padrões entre elementos de dados. O termo Mineração de Dados deveria ser utilizado para os estágios de descoberta do processo de *KDD*" .

Assim, *KDD* nada mais é do que realização da extração de possíveis conhecimentos úteis de dados, os quais se encontram ocultos. Teríamos a tendência de pensar, por esta definição, em uma nova técnica. Mas, na verdade, trata-se de um campo multi-disciplinar de pesquisa, envolvendo áreas como aprendizado de máquina, estatística, tecnologia de base de dados, sistemas especialistas e visualização de dados, os quais podem ser melhor observados na Figura 1.

Figura 1 - *KDD* é um campo multi-disciplinar (ADRIAANS; ZANTINGE, 1996)

2.2. Etapas do Processo de KDD

O processo de *KDD* consiste de três fases (MANNILA, 1996; FAMILI et al., 1997):

- **Pré-processamento:**

Consiste de todas as ações a serem realizadas antes do início do processo de análise dos dados. FAMILI et al. (1997), postulam que este processo deve ser feito entendendo a natureza dos dados, melhorando o desempenho na análise dos dados e, extraindo de maneira mais significativa os conhecimentos a partir dos dados. Essa etapa pode tomar até 80% do tempo necessário para todo o processo (MANNILA, 1996).

- **Mineração dos Dados:**

Envolve a aplicação de algoritmos específicos para a extração de padrões ou regras, a partir de um conjunto de dados com uma representação particular. A escolha do algoritmo a ser utilizado dependerá, fundamentalmente, do objetivo do processo *KDD*, sendo utilizadas técnicas da estatística e aprendizado de máquina, tais como aprendizado de regras, árvores de decisão, *clustering*, algoritmos genéticos e programação lógica, dentre outras (MANNILA, 1996).

- **Pós-processamento:**

Traduz os padrões descobertos em uma forma mais aceitável aos seres humanos. Isto pode ser feito através da visualização dos padrões extraídos (FAMILI et al., 1997).

2.3. Técnicas de Mineração de Dados

Segundo MANNILA (1996), *Data Mining* combina métodos e técnicas de, pelo menos, três áreas das anteriormente mencionadas na seção 2.1: aprendizado de máquina, estatística e banco de dados, sendo que as três áreas auxiliam no reconhecimento de regularidades, padrões ou conceitos, a partir de dados empíricos.

Existem diferentes classes de tarefas em mineração de dados, tais como: agrupamentos (*clusters*), classificação e regras de associação. Cada uma dessas classes tem como base um conjunto de algoritmos que serão utilizados na extração de relações relevantes dentro de uma massa de dados (CHEN; HAN; YU, 1996). Cada uma das classes de tarefas citadas, difere quanto ao tipo de problemas que o algoritmo é capaz de resolver. A seguir, descrevemos algumas dessas classes de tarefas, segundo LANGLEY (1996):

- **Agrupamentos:**

Esta proposta está, basicamente relacionada a problemas que envolvem a segmentação de dados, ou seja, a partir de um conjunto de dados, o sistema tem que dividir automaticamente os mesmos em grupos, baseados em suas similaridades e diferenças.

- **Classificação:**

O processo de classificação é uma técnica que consiste na aplicação de um conjunto de exemplos pré-classificados, para desenvolver um modelo capaz de classificar uma população maior de registros. Em geral, algoritmos de classificação incluem árvores de decisão, algoritmos genéticos ou redes neurais. Uma vez que o algoritmo classificador tenha sido desenvolvido de forma eficiente, ele será utilizado de forma preditiva para classificar novos registros naquelas mesmas classes pré-definidas.

- **Regras de Associação:**

O algoritmo de descoberta de regras de associação procura identificar afinidades entre registros de um subconjunto de dados. Essas afinidades são expressas na forma de regras: "60% de todos os registros que contém os itens A e B, também contém os itens C e D". A porcentagem de ocorrência representa um fator de confiança da regra, e costuma ser utilizada para eliminar tendências fracas, mantendo apenas as regras mais fortes.

2.4. Aprendizado de Máquina

Como comentado nas seções anteriores, umas das áreas relacionadas ao processo de mineração de dados é o aprendizado de máquina, sendo definido por ADRIAANS e ZANTINGE (1996), como um problema de busca, ou seja, encontrar um modelo formado por um conjunto de hipóteses (possíveis soluções) que representem o conjunto de dados (espaço de busca).

Quando nos referimos ao aprendizado de máquina relacionado ao processo de mineração de dados, todo este processo está focado na tarefa de classificação, a partir de observações já existentes (aprendizado indutivo).

De acordo com MARTÍNEZ-ENRÍQUEZ e ESCHALADA-IMAZ (1998), os sistemas de aprendizado indutivo podem ser classificados ou pela natureza da entrada de suas informações (forma de aprendizado), ou pela maneira com que seus sistemas realizam o processo de aprendizado. A seguir, é apresentado um breve resumo desses sistemas.

- **Aprendizado não-supervisionado:**

Cada observação dentro de um conjunto é descrita pelo mesmo conjunto de atributos, e isto forma a natureza da informação de entrada. O resultado é um conjunto de agrupamentos (*clusters*), individualmente definidos por um conceito diferente e particular.

- **Aprendizado supervisionado:**

Os agrupamentos já são fornecidos pelo sistema e a função do sistema é criar um classificador que classifique as observações, através da apresentação de sucessivas observações, verificando se as mesmas foram classificadas corretamente se auto-ajustando.

- **Aprendizado não-incremental:**

O sistema constrói classificadores utilizando-se de todo o conjunto de observações simultanemantes para alcançar a solução final. Desta forma, nestes sistemas não-incrementais, é possível determinar a complexidade de seus algoritmos, sendo útil em

aplicações reais. Contudo, se considerarmos uma nova observação dentro do sistema, é necessário processar novamente todo conjunto de observações.

- **Aprendizado incremental:**

O sistema constrói classificadores e os refina num processo interativo, observação por observação. Assim, o processo altera alguns agrupamentos à medida que cada nova observação é manuseada. Desta forma, os sistemas incrementais podem trabalhar individualmente com novas observações, em qualquer tempo.

Dentre as várias técnicas existentes para o processo de aprendizado indutivo, a área de aprendizado de máquina pode ser dividida em 5 paradigmas segundo LANGLEY (1996):

- **Redes Neurais:**

As Redes Neurais representam o conhecimento através de uma rede de multi-camadas de unidades de processamento, ativadas a partir de nodos de entrada, percorrendo as unidades até nodos de saída. Pesos nas ligações determinam o quanto de ativação é transmitida em cada caso. As ativações nos nodos de saída podem ser traduzidas em prognósticos numéricos ou decisões discretas sobre a classe de entrada. A preocupação com as redes neurais é melhorar a precisão de classificação através da modificação dos pesos de ligação. Um algoritmo típico de aprendizado realiza uma busca do tipo *hill-climbing,* através do espaço de pesos, modificando-os no sentido de minimizar os erros que a rede faz com os dados de treinamento. Aplicações utilizando redes neurais para o processo de classificação podem ser encontradas em KOHONEN (1995) e STEINER (1995).

- **Aprendizado baseado em casos ou instâncias:**

Representam o conhecimento a partir de casos específicos ou instâncias (exemplos) com classificação conhecida para classificar novos dados através de comparação de similaridades. O caso-exemplo mais parecido com o dado a ser classificado, segundo algum critério de avaliação, indicará a qual classe pertence este exemplo (AHA; KLIBER; ALBERT, 1991).

- **Algoritmos Genéticos:**

Sendo um dos ramos da Computação Evolucionária, os Algoritmos Genéticos são inspirados na Teoria da Evolução. Representam o conhecimento como a "carga genética" de um "indivíduo". Um conjunto de soluções possíveis ou indivíduos, chamado "população", é analisado segundo algum critério (função de aptidão ou "fitness"), e novas soluções são criadas e tentadas usando como base o conjunto anterior, através de operadores de recombinação (*crossover*) e "mutação". A população é continuamente melhorada pela seleção dos indivíduos mais bem adaptados, e o processo de avaliação-seleção-cruzamento-mutação é repetido até que algum critério de término seja atingido (MITCHELL, 1997).

- **Indução de Regras:**

O paradigma de indução de regras trabalha, particularmente com regras condição-ação, "se <condição atendida> então <classe x>", árvores de decisão (QUINLAN, 1993), ou alguma estrutura similar de conhecimento (HOLSHEIMER; KERSTEN; SIEBES, 1996; CLARK; NIBLETT, 1989). Algoritmos de aprendizado em estruturas de indução de regras, geralmente realizam uma busca *greedy* através do espaço indução, usando uma função de avaliação estatística para selecionar atributos e, incorporá-los na sua estrutura de conhecimento. A maioria dos métodos divide o conjunto de dados de treinamento recursivamente, em conjuntos separados, preocupando-se em sumarizar cada conjunto como condições lógicas de conjunções (LANGLEY, 1996).

- **Aprendizado Analítico:**

Representa o conhecimento como regras na forma lógica, usando um método de busca para resolver várias fases do problema. As técnicas comuns representam seu conhecimento como regras de inferência. O mecanismo de aprendizado utiliza um conhecimento *background* para construir provas ou explicações da experiência e então compila as provas em regras mais complexas, solucionando problemas similares ou com menos busca ou, em uma única etapa. A maior parte do problema no aprendizado analítico se preocupa com o melhoramento do processo de busca, mas alguns vem

trabalhando com o aprimoramento da precisão nas tarefas de classificação (LANGLEY, 1996).

Apesar de toda a relevância destas áreas de pesquisa em aprendizado de máquina, optamos por centrar o estudo do presente trabalho em Algoritmos Genéticos. Esta área de estudo, tem merecido grande atenção por parte de vários pesquisadores devido a sua grande capacidade de lidar com dados inválidos ou imprecisos, se comparada a outras técnicas.

Assim, será apresentada no capítulo seguinte, uma discussão mais detalhada sobre os conceitos e estrutura de funcionamento dos mesmos.

3. Algoritmos Genéticos

Algoritmos Genéticos são métodos baseados em processos genéticos de um organismo biológico, inspirados na Teoria da Evolução de Charles Darwin, cuja principal finalidade refere-se à solução de problemas de busca e otimização.

Os princípios básicos dos Algoritmos Genéticos foram primeiramente descritos por HOLLAND (1975), e, posteriormente, por outros autores como GREFENSTETTE (1986), DAVIS (1987), GOLDBERG (1989), GREFENSTETTE (1990), DAVIS (1991) e MITCHELL (1997).

Esses algoritmos simulam processos naturais de sobrevivência e reprodução das populações, essenciais em sua evolução. Na natureza, indivíduos de uma mesma população competem entre si, buscando principalmente a sobrevivência, seja através da busca de recursos como alimento, ou simplesmente o encontro de outro indivíduo da mesma espécie, visando a reprodução. Os indivíduos mais aptos terão um maior número de descendentes, ao contrário dos indivíduos menos aptos.

Da mesma forma, genes altamente adaptados ou indivíduos adaptados (*fit*), irão produzir um maior número de descendentes a cada geração. A combinação de boas características de diferentes ancestrais pode, muitas vezes, produzir descendentes mais bem adaptados que seus "pais", ou seja, cuja *fitness* é melhor que a *fitness* de seus "pais", tornado-os mais adaptados ao seu ambiente.

Os algoritmos genéticos usam exatamente esta analogia. Trabalhando com populações de indivíduos, cada um representando uma possível solução de um problema, indivíduos são selecionados, baseados em um índice que representa o quanto cada um destes é uma boa solução para o problema (*fitness score*). A este novo conjunto é dada então a possibilidade de reproduzir-se, gerando novos indivíduos como descendentes, que compartilham partes de suas características oriundas de cada um de seus "pais".

O processo é então repetido, selecionando novos indivíduos a partir da nova geração e, aplicando-se novamente operações de cruzamento entre os mesmos. Quando satisfeito algum critério estabelecido, o processo é finalizado, gerando um conjunto de indivíduos ou possíveis soluções para o problema.

3.1. Princípios Básicos

Um AG padrão está esquematizado nas Figuras 2 e 3. A idéia básica do funcionamento de um algoritmo genético é tratar possíveis soluções de um problema como "indivíduos' de uma "população", que irá evoluir a cada iteração ou "geração" (HASSE, 2000). Porém, antes de iniciar o AG, uma codificação satisfatória (ou representação) do problema deve ser realizada, armazenando os dados a serem otimizados.

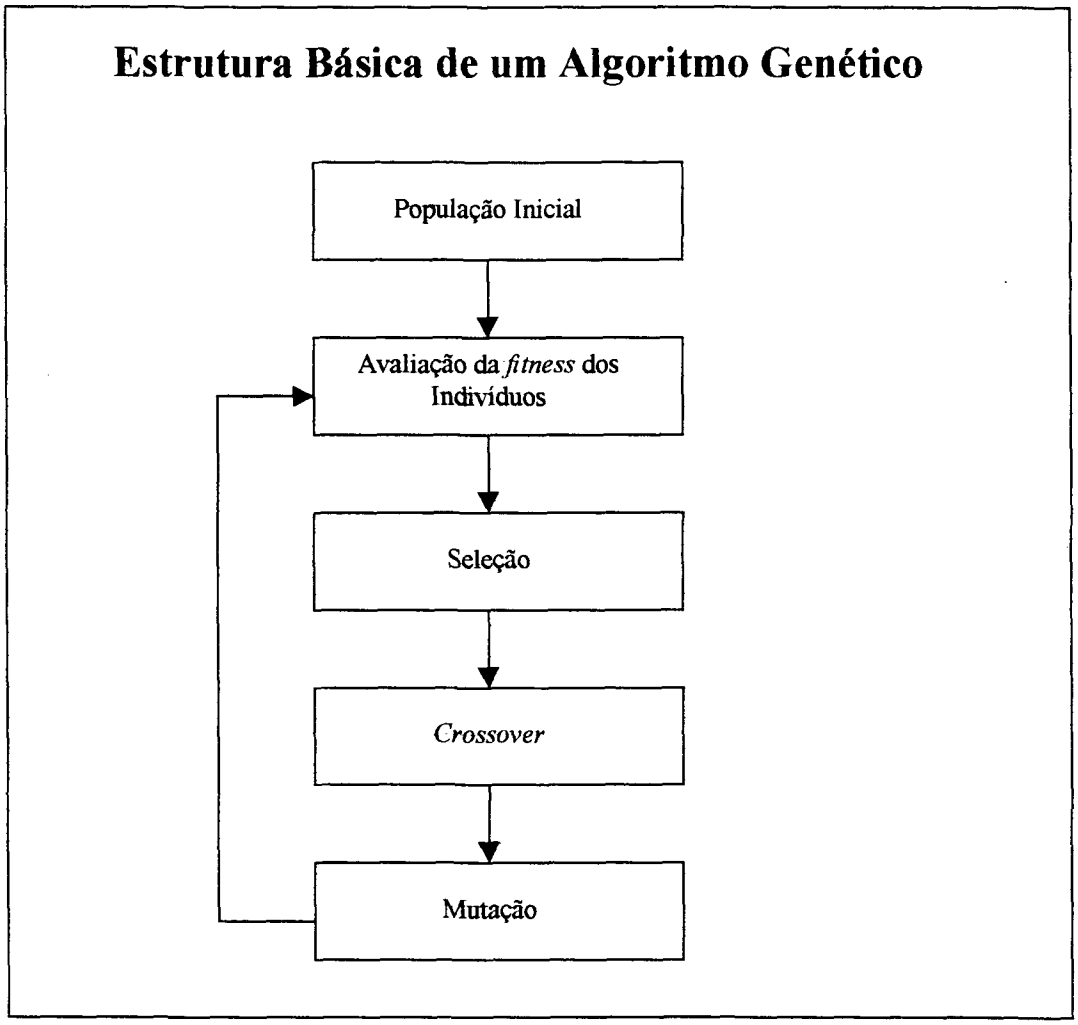

Figura 2 - Estrutura Básica de um Algoritmo Genético

```
Início
 Gerar um população inicial
 Calcular a fitness de cada indivíduo
 Enquanto não finalizado faça
 Início
        Conjunto nova população igual a vazio
        Para (tamanho_população / 2 ) faça
        Início
                Selecionar dois indivíduos a partir da população anterior
                ( em favor de um valor de fitness)
                Recombinar os dois indivíduos para obter dois descendentes
                Inserir descendentes na nova geração
        Fim
        Calcular fitness dos indivíduos da nova população
        Faça população igual a nova população
        Se população convergiu então
                Finalizado = Verdade
 Fim
Fim
```

Figura 3 - Um algoritmo genético clássico

Assim, a primeira etapa do processo é a escolha de uma população inicial, ou semente, geralmente formada por indivíduos selecionados aleatoriamente. Em uma etapa seguinte, torna-se necessária uma função de avaliação ou função *fitness*. Esta função indicará o mérito de cada solução codificada que, associada a um método específico de seleção, ou operador de seleção, formando um conjunto de possíveis soluções, chamada também de nova geração.

A partir desta nova geração, será iniciado o processo de cruzamento, onde pares de indivíduos ("pais"), terão suas características ("genes") recombinadas, através de um processo conhecido como *crossover*. Novamente, imitando o processo de reprodução natural, um único indivíduo poderá ter um de seus "genes" modificados aleatoriamente, imitando o processo de mutação que algumas vezes ocorre na natureza.

Estes passos são então repetidos várias vezes, podendo ter como critério de parada, um número pré-determinado de iterações (gerações), ou até que uma solução aceitável seja encontrada.

3.1.1. Codificação

A codificação de cada indivíduo na realidade é composta apenas do "material genético", ou seja, o conjunto de atributos que pode ser considerado como uma das soluções possíveis para o problema analisado (HASSE, 2000).

A principal dificuldade no tratamento dos indivíduos é a forma de representar e organizar os dados. Segundo MITCHELL (1997), nos últimos anos, vários métodos experimentais de codificação têm sido utilizados, sendo alguns apresentados nos itens 3.1.1.1 e 3.1.1.2.

3.1.1.1. Codificação Binária

É uma das formas mais comuns de codificação, principalmente devido a um fator histórico, já que os primeiros trabalhos de AGs desenvolvidos por HOLLAND (1975), e outros pesquisadores, se concentraram nesta forma de codificação.

Na codificação binária, cada atributo é representado como uma seqüência de bits (0 ou 1) e o indivíduo, como a concatenação das seqüências de bits de todas as suas características.

Outras variações, a partir da forma original de codificação binária, podem ser encontradas em HOLLAND (1975), BETHKE (1980), CARUANA e SCHAFFER (1988) e GOLDBERG (1989).

Codificações binárias possuem a vantagem de serem simples de implementar e fáceis para a adaptação a qualquer novo operador genético. Entretanto, existem algumas desvantagens, como uma boa forma de codificação/decodificação dos bits, podendo comprometer o desempenho do algoritmo. Outra desvantagem refere-se ao processo de cruzamento e mutação, pois devido à característica de codificação, existe a possibilidade de gerar indivíduos que não sejam válidos, devendo-se assim, desenvolver processos que se preocupem em validar os indivíduos gerados.

3.1.1.2. Codificação através de vários caracteres ou valores reais

Outra forma de representação muito utilizada é a codificação através do próprio alfabeto de características que se deseja representar (letras, códigos, números reais, etc). Alguns exemplos podem ser encontrados em MONTANA e DAVIS (1989), MEYER e PACKARD (1992), SCHULZE-KREMER (1992), KITANO (1994).

A principal vantagem desse sistema de codificação é a sua simplicidade de entendimento e manipulação, apesar da implementação do algoritmo genético e seus operadores torná-lo um pouco mais trabalhoso pela adaptação do novo algoritmo à nova forma de codificação.

Outras formas são possíveis, dependendo do tipo de problema. De qualquer modo, ainda há uma grande polêmica quanto ao melhor sistema de codificação (MITCHELL, 1997).

3.1.2. Função *fitness*

Cada indivíduo de uma população tem associado a si, um valor de aptidão (*fit*), identificando seu grau de adaptabilidade ao ambiente (MITCHELL, 1997). A função *fitness* ou de avaliação, é um dos principais componentes de um algoritmo genético, medindo a proximidade de cada indivíduo à solução desejada, ou sua qualidade com relação à solução, se comparada a outra solução (indivíduo) da população.

Desta maneira, os indivíduos da população corrente devem ser avaliados individualmente pela função *fitness*, verificando se os mesmos estão ou não aptos a solucionar o problema (ou próximos da solução).

Representando o grau de adaptação de cada indivíduo ao meio ambiente, quanto mais alto for o valor de aptidão de uma solução, maior será a capacidade de sobrevivência e reprodução e, consequentemente, maior sua representatividade na próxima geração.

Esta representatividade é essencial ao algoritmo, discriminando corretamente a proporção de boas soluções em relação às más. A ocorrência de baixa representatividade,

ou seja, funções *fitness* que apresentam pouca precisão, muitas vezes levam ao descarte de soluções promissoras durante a execução do algoritmo e à exploração do espaço de busca em direções não muito promitentes.

A relação entre precisão e tempo de execução da função *fitness* é outro fator importante a analisar na implementação de um algoritmo. Funções extremamente precisas, geralmente possuem um custo computacional elevado, dificultando seu uso em grandes populações. Por outro lado, funções mais simples, com menor grau de precisão, permitem maior exploração do espaço de busca sem elevar o custo computacional, porém tem a desvantagem de muitas vezes desconsiderar possibilidades promissoras ao alcance de uma solução melhor.

Como ilustração da função *fitness*, é apresentada na Tabela 1 um exemplo simples, onde o valor da função é determinado simplesmente pela quantidade de bits "1" dos indivíduos.

Tabela 1 - Exemplo de População e Função de Ajuste (MITCHELL,1997)

Indivíduos	**Função Ajuste**	**Proporção em relação ao total da população**
00000110	2	16,67%
11101110	6	50,00%
00100000	1	8,33%
00110100	3	25,00%
Total	12	100,00%

3.1.3. Reprodução: Seleção, Cruzamento e Mutação

Durante a fase de reprodução dos AGs, indivíduos são selecionados a partir de uma população e, recombinados, produzindo descendentes que serão incluídos na próxima geração. "Pais" são selecionados aleatoriamente, a partir da população, utilizando-se de uma forma que favoreça indivíduos com um maior valor de função de ajuste ou *fitness*.

Assim, bons indivíduos provavelmente serão selecionados de uma vez em uma geração, enquanto que indivíduos com menor *fitness*, não serão necessariamente selecionados.

Selecionando-se dois "pais", seus cromossomos serão recombinados, utilizando os mecanismos típicos de cruzamento (*crossover*) e mutação. As formas mais básicas para este processo de reprodução são seleção, *crossover* e mutação.

3.1.3.1. Seleção

O método de seleção escolhe quais indivíduos participarão efetivamente na criação da próxima geração da população, tendo como propósito, aumentar a probabilidade de seleção de indivíduos com maior valor de aptidão para o processo de reprodução (DAVIS, 1991).

Contudo, principalmente nas primeiras gerações, não se deve enfatizar excessivamente a seleção apenas dos melhores indivíduos. Isso faz com que a diversidade seja rapidamente perdida, levando a encontrar apenas soluções ótimas locais.

Segundo MITCHELL (1997), podemos citar entre alguns dos métodos de seleção:

- **Roleta**

É conhecida como seleção proporcional ao valor de *fitness*. Neste método, cada indivíduo da população é representado em uma fatia de uma roleta circular proporcionalmente ao seu índice de aptidão, como é apresentado na Figura 4,. Assim, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa é dada uma porção relativamente menor. Finalmente, a roleta é girada, sorteando o indivíduo que irá participar da próxima geração. Este processo é repetido N vezes, onde N representa o número de indivíduos da população.

Segundo MITCHELL (1997), o método da roleta pode ser implementado da seguinte maneira:

1. Somar os valores das funções de ajuste de todos os indivíduos da população, chamadas soma T.
2. Repetir N vezes para selecionar N indivíduos.

3. Escolher um número aleatório inteiro, r, entre 0 e T.
4. Percorrer através dos indivíduos da população, calculando a soma acumulada da função de ajuste dos indivíduos já percorridos, até alcançar uma soma maior ou igual a r. O indivíduo com o valor neste limite será selecionado.

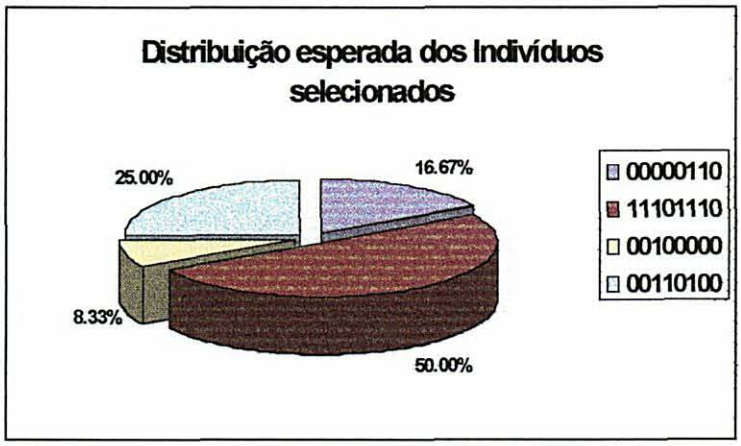

Figura 4 – Seleção por "roleta" (dados da Tabela 1)

Este método resulta estatisticamente em uma distribuição proporcional de descendentes para cada indivíduo; contudo, em populações relativamente pequenas, pode acabar resultando em péssimas escolhas. Mesmo que na média espere-se uma distribuição proporcional à função de ajuste, em uma determinada geração, pode ocorrer a escolha do mesmo indivíduo diversas vezes ou ainda, a omissão de um ótimo indivíduo por ele não ter sido "sorteado".

- ***Stochastic Universal Sampling***

No sentido de amenizar o efeito de escolha do mesmo indivíduo diversas vezes pelo método da "roleta", encontramos em BAKER (1987), a proposta de uma variação desse método, o *Stochastic Universal Sampling*. No *Stochastic Universal Sampling*, escolhe-se apenas um número aleatório que servirá de base para todas as amostragens, garantindo uma distribuição o mais proporcional possível à função de ajuste de cada indivíduo.

De qualquer forma, os métodos baseados na seleção proporcional ao valor de *fitness* geralmente não solucionam todos os tipos de problemas.No início da busca, a variância da *fitness* é alta, e o número de indivíduos mais bem adaptados é pequeno. Como uma das características deste método de seleção é beneficiar com maior freqüência os indivíduos mais bem adaptados, seus descendentes se espalham rapidamente por toda a população, principalmente nas primeiras gerações, levando o algoritmo a soluções ótimas locais.

À medida que o algoritmo genético se aproxima do final do processo de busca, a variância é quase nula e a seleção torna-se pouco eficiente.

- **Elitismo**

Introduzido primeiramente por De JONG (1975), o elitismo é uma estratégia utilizada nos métodos de seleção para forçar o algoritmo a reter um certo número dos melhores indivíduos em cada geração. Tais indivíduos podem ser perdidos se eles não forem selecionados para reprodução ou se eles forem destruídos por cruzamento (*crossover*) ou mutação. Através dessa estratégia, muitos pesquisadores têm conseguido melhoramentos significativos no desempenho dos AGs (MITCHELL, 1997).

- **Torneio**

O processo utilizado no método de torneio, apresentado esquematicamente na Figura 5, é extremamente simples. Inicialmente, selecionam-se aleatoriamente dois indivíduos. Em seguida, é escolhido um número aleatório entre 0 e 1, o qual é comparado com um parâmetro K, pré-definido pelo usuário (geralmente 0,75). Caso o número escolhido seja menor que K, o melhor dos dois indivíduos será selecionado, caso contrário, será escolhido o pior indivíduo. Estes passos são repetidos até selecionar todos os indivíduos necessários para formar a população.

A principal característica desse método é a sua convergência lenta e constante, facilitando uma boa exploração do espaço de busca, evitando uma busca improdutiva, principalmente se considerados domínios complexos com a presença de um número grande de pontos ótimos locais. Análises deste método podem ser encontradas em GOLDBERG e DEB (1991).

1. Escolhe-se aleatoriamente dois indivíduos.
2. Escolhe-se um número aleatório r entre 0 e 1. O seu valor é comparado com um parâmetro k (próximo ao 0.75, por exemplo).
3. Se r for menor que k, o melhor dos indivíduos é escolhido, caso contrário é escolhido o pior.
4. Retorna-se ao passo 1 até ter sido selecionado um número suficiente de indivíduos.

Figura 5 – Possível implementação do método de seleção por torneio (MITCHELL, 1997)

3.1.3.2. Cruzamento (*crossover*)

Considerado como um dos operadores no processo de reprodução, a função do operador de cruzamento, conhecido também como *crossover*, é a criação de novos indivíduos através da mistura das características de seus "pais". Este processo procura imitar o procedimento de reprodução em células.

Existem diversas maneiras de se fazer o cruzamento, sem nenhum consenso sobre qual funciona melhor em cada tipo de problema, pois o sucesso deste operador estará intimamente ligado à função *fitness*, sistema de codificação e outros detalhes do algoritmo (MITCHELL, 1997). A seguir, algumas das formas de cruzamento são apresentadas.

- **Cruzamento simples ou de um ponto**

Em um processo de cruzamento simples, selecionados dois indivíduos, suas *strings* de cromossomos são divididas em alguma posição escolhida aleatoriamente, produzindo dois segmentos, cabeça e cauda. Os dois segmentos de cauda são então trocados, formando novamente dois novos cromossomos completos (Figura 6), tendo os dois descendentes

herdado uma parte dos genes da cada "pai". O resultado desta operação é um indivíduo que potencialmente combina as melhores características dos indivíduos usados como base.

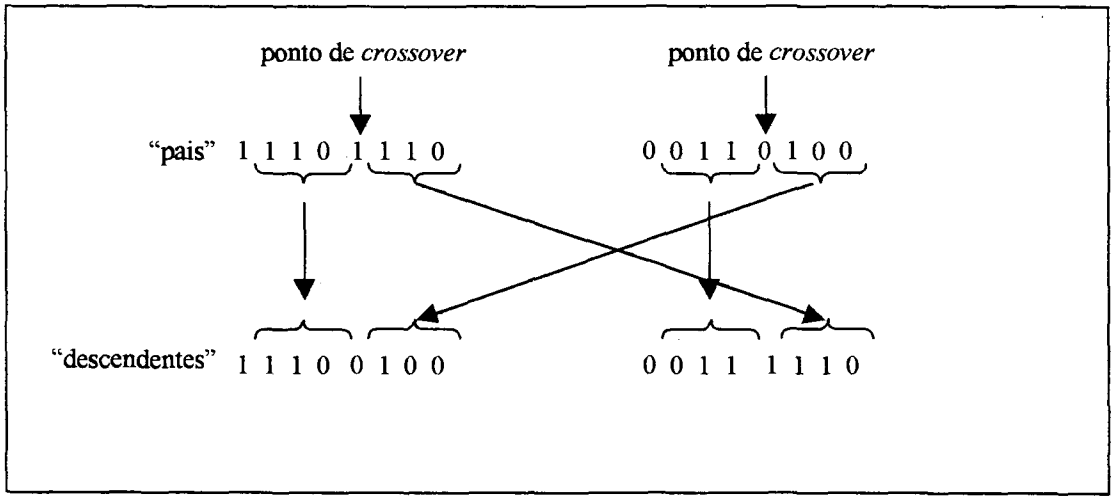

Figura 6 - Cruzamento em um ponto

- **Dois pontos de cruzamento (*two-point crossover*)**

Procede-se de maneira similar ao cruzamento de um ponto, mas a troca de segmentos do gene ocorre como mostrado na Figura 7, na qual a termo de representação, foram escolhidos os pontos 2 e 6.

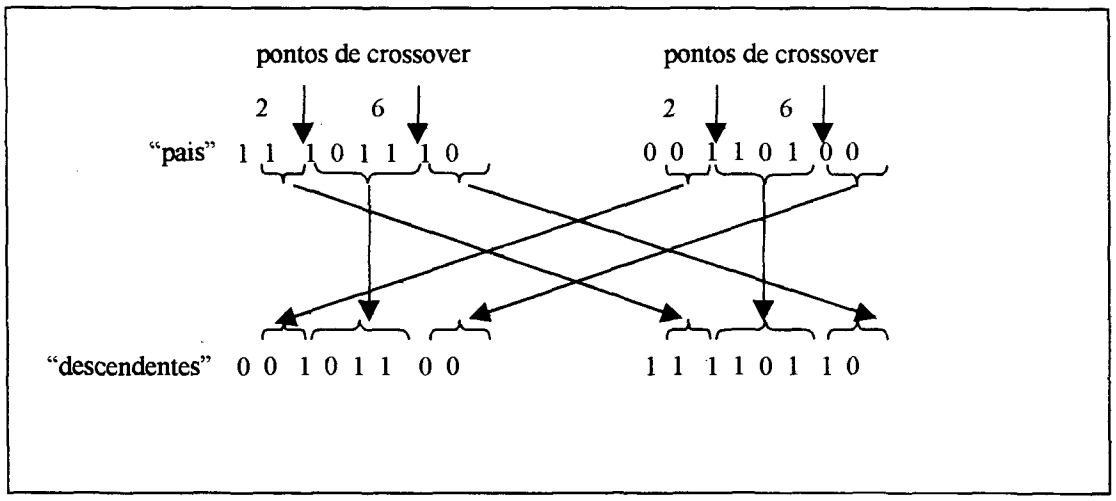

Figura 7 - Cruzamento em dois pontos

- **Cruzamento Uniforme (*uniform crossover*)**

Este tipo de cruzamento é radicalmente diferente dos cruzamentos anteriores. Cada gene do descendente é criado copiando o gene correspondente de um dos "pais", escolhido de acordo com uma máscara de cruzamento gerada aleatoriamente. Onde houver 1 na máscara de cruzamento, o gene correspondente será copiado do primeiro "pai" e, onde houver 0 será copiado do segundo. O processo é repetido com os "pais" trocados para produzir o segundo descendente. Uma nova máscara de cruzamento é criada para cada par de "pais". A Figura 8 demonstra graficamente este processo.

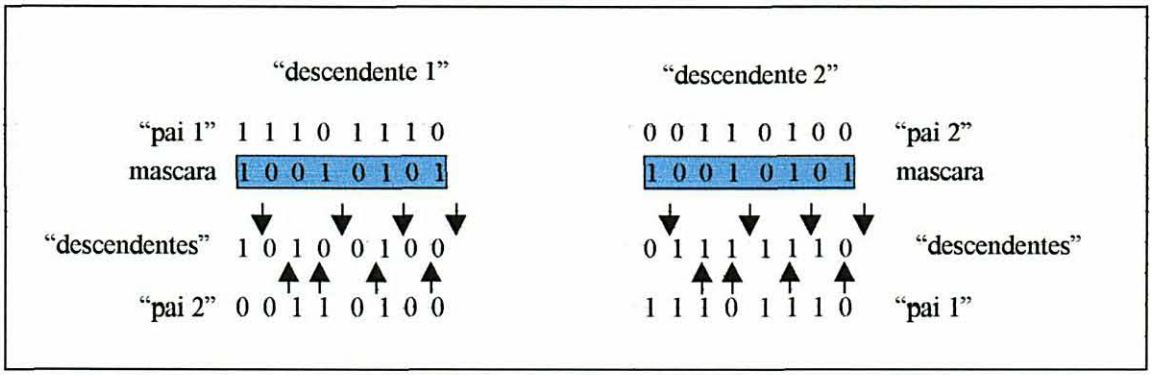

Figura 8 - Cruzamento uniforme

É importante notar que o processo de cruzamento não é aplicado a todos os pares de indivíduos selecionados. Uma escolha aleatória é realizada, onde há uma probabilidade de se aplicar *crossover* a um par de indivíduos. Caso o *crossover* não seja aplicado, descendentes serão produzidos simplesmente pela duplicação de seus "pais".

Existem também algumas formas de melhorar o cruzamento, dentre elas, a restrição de quais pares de indivíduos podem ser utilizados como "pais". Esta restrição pode ser realizada impedindo-se o cruzamento de indivíduos muito similares, evitando o "incesto". Um outro modo pode ser restringindo o cruzamento de indivíduos muito diferentes, fazendo com que indivíduos próximos um do outro no espaço de busca, tendam a se cruzar e convergir mais rapidamente.

Ao longo de diversas gerações, esta operação acabará por encontrar combinações que aumentem o valor da função *fitness* acima dos valores de qualquer um dos dois indivíduos originais. A operação de "seleção" usará com mais freqüência os indivíduos

mais bem adaptados e, devido a isso, a média do valor da função de ajuste da população tenderá a aumentar lentamente.

3.1.3.3. Mutação

Após a geração de um descendente, aplicado ou não o operador de *crossover*, o processo de mutação representa uma pequena probabilidade de alteração aleatória de um gene, geralmente em torno de 0,01 (MITCHELL, 1997). A Figura 9, exibe o quinto gene de um cromossomo sendo mutado.

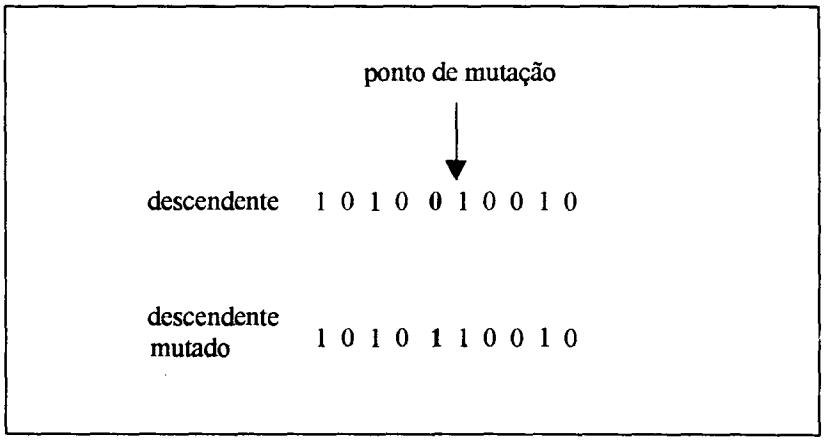

Figura 9 - Mutação simples

Do ponto de vista tradicional, o processo de *crossover* é o mais importante dentre as técnicas de mutação e recombinação, pois permite uma rápida exploração do espaço de busca (HOLLAND, 1975). A mutação representa uma pequena parcela durante o processo de busca; sua função principal é permitir que outros pontos de busca possam ser visitados, evitando assim a obtenção de pontos ótimos locais.

3.1.4. Considerações Finais sobre AGs

Da mesma forma que ocorre nos processos de reprodução natural, os operadores de seleção, mutação e cruzamento levam tempo para mostrar de forma perceptível seus efeitos. Se o número de gerações for muito pequeno, o algoritmo não terá tempo para "evoluir" até chegar às melhores soluções. Por outro lado, se o número de gerações for muito grande, o

algoritmo, muitas vezes, apresentará pouca ou nenhuma melhora para o processo de busca, além do alto dispêndio computacional.

Podemos então perceber a importância fundamental no estudo do número de gerações, assim como os outros parâmetros descritos anteriormente. Esses conjuntos de parâmetros são cruciais na especialização dos Algoritmos Genéticos para o alcance de melhores resultados, revertendo em ganhos para diferentes campos de aplicação, tais como biologia molecular, projetos de circuitos digitais, tarefas que envolvam otimização e mineração de dados (MITCHELL, 1997).

O próximo capítulo refere-se à utilização de Algoritmos Genéticos em Mineração de Dados e suas possíveis especializações.

4. Algoritmos Genéticos no contexto de Mineração de Dados

Algoritmos Genéticos são utilizados para resolver praticamente qualquer tipo de problema que envolva a exploração de soluções num espaço de busca. Para implementá-los, basta ter uma representação da solução candidata e uma função de avaliação que seja razoavelmente precisa.

Exemplos de aplicações que usam algoritmos genéticos incluem problemas de otimização de soluções, aprendizado de máquina, desenvolvimento de estratégias e fórmulas matemáticas, análise de modelos econômicos, problemas de engenharia, diversas aplicações na Biologia como simulação de bactérias, sistemas imunológicos, ecossistemas, descoberta de formato e propriedades de moléculas orgânicas (MITCHELL, 1997).

Devido ao grande interesse por diferentes setores, no sentido de extrair informações úteis em bases de dados e utilizá-las de forma estratégica na tomada de decisões, o processo de Mineração de Dados utilizando AGs surge com grande potencial.

Segundo FREITAS e LAVINGTON (1998), existem diferentes áreas de aplicação que utilizam um Algoritmo Genético:

- Buscar regras no formato SE-ENTÃO em conjuntos de dados, tais como regras de classificação, onde o antecedente (parte "SE") contém condições de valores de atributos e o consequente (parte "ENTÃO") contém a classe prevista quando as condições do antecedente forem satisfeitas. O algoritmo genético usa regras neste formato como indivíduos da população. A função de aptidão será calculada pela qualidade da regra gerada (FREITAS; LAVINGTON, 1998).

- Cada indivíduo pode ser também um conjunto de "protótipos" para fins de classificação. Neste caso, a tarefa do Algoritmo Genético consiste em encontrar o melhor conjunto de protótipos de acordo com o grau de acerto deste conjunto. Este paradigma de classificação é baseado por semelhança entre instâncias, *ou Instance-Based Learning (IBL)*. A implementação deste tipo de Algoritmo Genético pode ser encontrada em KNIGHT e SEM (1995), enquanto o paradigma IBL é abordado em detalhes em AHA, KLIBER e ALBERT (1991).

- Algoritmos Genéticos também podem ser utilizados de forma indireta, na determinação da importância de cada atributo para fins de classificação em um algoritmo como IBL (KELLY e DAVIS, 1991).

- Outras diferentes abordagens de aplicação ou variações das formas citadas anteriormente podem ser encontradas em FREITAS e LAVINGTON (1998).

Dentro das abordagens de aplicação dos Algoritmos Genéticos, no contexto de Mineração de Dados, o presente trabalho foca o chamado processo de classificação onde, a partir de um conjunto de exemplos previamente classificados, chamado conjunto de treinamento, será realizada uma análise dos mesmos, criando uma descrição lógica, capaz de classificar corretamente novos exemplos de classe desconhecida. Neste caso, a descrição lógica gerada poderá ser considerada um classificador.

Para uma melhor visualização da tarefa de classificação, a Tabela 2 e Figura 10 mostram respectivamente os dados de entradas para um sistema de classificação e o conjunto de regras geradas por este sistema (FREITAS; LAVINGTON, 1998).

Tabela 2 – Dados de entrada para um sistema de classificação (FREITAS e LAVINGTON, 1998)

Sexo	**País**	**Idade**	**Comprar (meta)**
M	França	25	Sim
M	Inglaterra	21	Sim
F	França	23	Sim
F	Inglaterra	34	Sim
F	França	30	Não
M	Alemanha	21	Não
M	Alemanha	20	Não
F	Alemanha	18	Não
F	França	34	não
M	França	55	não

SE (País = "Alemanha") ENTÃO (Comprar = "não")
SE (País = "Inglaterra") ENTÃO (Comprar = "sim")
SE (País = "França" E Idade $\leq$ 25) ENTÃO (Comprar = "sim")
SE (País = "França" E Idade > 25) ENTÃO (Comprar = "não")

Figura 10 – Regras de classificação descobertas a partir dos dados da Tabela 2 (FREITAS e LAVINGTON, 1998)

4.1. Abordagens e Heurísticas utilizadas em Algoritmos Genéticos

No processo de indução de regras através do uso de Algoritmos Genéticos, existem duas abordagens significativamente diferentes: abordagem de Pittsburgh (SMITH, 1983) e a abordagem de Michigan (HOLLAND, 1986).

Na abordagem de Pittsburgh, cada indivíduo da população contém um classificador completo, tornando o algoritmo simples, não havendo necessidade de criar heurísticas ou formação de nichos no processo de criação do conjuntos de regras. A principal desvantagem desta abordagem, refere-se à grande redundância dentro da população, que pode levar à necessidade da criação de grandes populações e indivíduos com uma codificação muito grande (GIORDANA; NERI, 1995).

Em contrapartida, na abordagem de Michigan, cada indivíduo da população corresponde a uma única regra simples, classificando apenas um subconjunto dos dados de toda a população. Nesta abordagem, torna-se necessário desenvolver algum tipo de estratégia para extrair da população um conjunto de regras não redundantes (HOLLAND, 1986; WILSON, 1987). Uma das formas de extrair um conjunto de regras não redundantes é a formação de "nichos". Técnicas de formação de nichos (WATSON, 1998) forçam a competição entre indivíduos similares, criando assim sub-populações que exploram regiões potencialmente distintas do espaço de busca.

Alguma técnica de formação de nichos deve ser usada a fim de explorar e manter a diversidade dentro da população. A maioria dos métodos de formação de nichos é baseada em *Crowding* (De JONG, 1975; MAHFOUD, 1992) ou em Compartilhamento de Recursos, *Sharing* (GOLDBERG; RICHARDSON, 1987; DEB; GOLDBERG 1989).

Quando se usa *Crowding* para criar nichos, a substituição de indivíduos na população é modificada para fazer com que novos indivíduos substituam outros similares com menor valor para a função de aptidão dentro da população. Neste caso, a seleção e demais operações como cruzamento e mutação não são modificadas. De acordo com os testes feitos por WATSON (1998), este método costuma chegar mais próximo dos valores

ótimos locais, mas não se comporta tão bem quanto o *sharing* para manter sub-populações estáveis de indivíduos.

Com relação ao compartilhamento de recursos, a formação de nichos ocorre através de redução do valor da função de avaliação, proporcionalmente à presença de indivíduos similares dentro da população, forçando uma diversidade no processo de seleção de indivíduos. A similaridade entre indivíduos pode ser medida tanto no espaço genotípico quanto no espaço fenotípico (MITCHELL, 1997).

O espaço genotípico compreende a semelhança entre as codificações dos indivíduos, medida, por exemplo, pela quantidade de bits iguais entre eles. Medir a similaridade entre regras no espaço genotípico não parece ser uma boa alternativa para um classificador, já que é possível, e muito provável, que possam existir regras muito diferentes quanto à codificação e atributos usados que cubram exatamente os mesmos exemplos de treinamento. Segundo GIORDANA e NERI (1995), não existe nenhuma métrica geral de distância entre duas regras que seja satisfatória para a tarefa de classificação.

O compartilhamento de recursos no espaço fenotípico permite uma comparação mais adequada de similaridade. Isto ocorre porque a similaridade é medida pelo número de exemplos de treinamento que são cobertos simultaneamente pelas regras. Quanto mais exemplos iguais duas regras cobrem, mais a função de aptidão delas será reduzida.

Algumas das aplicações utilizando o conceito de criação de nichos, podem ser encontradas em McCALLUM e SPACKMAN (1990), HORN, DEB e GOLDEBERG (1994), GIORDANA e NERI (1995) e HASSE (2000).

Além disso, outros Algoritmos Genéticos híbridos ou Algoritmos Genéticos integrados a *simulated annealing*, busca Tabu (*tabu search*) e outras heurísticas têm sido estudados para melhorar o desempenho dos Ags na solução de problemas complexos (KURAHASHI; TERANO, 2000).

Assim, nas próximas seções, comentaremos sobre o uso de busca Tabu e uma proposta de utilização da mesma integrada a um Algoritmo Genético.

4.2. Busca Tabu

O método de busca Tabu é um procedimento heurístico proposto por GLOVER (1986) para resolver problemas de otimização combinatória. A idéia básica é evitar que a busca por soluções ótimas termine por encontrar um mínimo local.

A filosofia de busca Tabu (BT) é derivada e explorada por um conjunto de princípios da solução de problemas inteligentes. O procedimento dessa meta-heurística foi projetado para realizar sua busca através de soluções vizinhas, para problemas de otimização combinatória. Assim como outros métodos tradicionais, busca Tabu necessita utilizar em seu procedimento um grande domínio de conhecimento (YONG, 1994).

Alguns elementos básicos em uma implementação de uma BT são:

- **Movimentos**

Um movimento representa uma função que transforma uma solução em outra, ou uma função vizinha.

- **Lista Tabu**

Esta lista, também conhecida por lista de restrições, refere-se a uma memória de todas as soluções anteriores durante o processo de busca do algoritmo. O tamanho da Lista Tabu e os operadores que modificam a lista são importantes regras no desempenho da busca Tabu. Estes operadores são responsáveis pela adição de um novo movimento, a lista de restrições, e como e quando um movimento deve ser removido da Lista Tabu.

- **Critério de Aspiração**

O critério de aspiração funciona como uma forma de determinar quando uma restrição Tabu deve ser ignorada, realizando o movimento independente se classificado como proibido. O uso adequado deste critério pode ser muito importante para que a busca Tabu alcance melhores níveis de desempenho.

- **Término**

O processo de busca Tabu deve ser finalizado quando não existir mais nenhum movimento possível a ser realizado ou quando alcançar o número máximo de iterações definidas pelo usuário.

- **Definição de Parâmetros**

Como a busca Tabu é orientada ao problema, ela requer um ajuste fino de certos parâmetros, tais como regras de parada, solução inicial, tamanho de Lista Tabu, número máximo de iterações, e nível de aspiração.

Um esquema geral do processo de busca Tabu é ilustrado na Figura 11. Segundo HORNE e MACBETH (1998), no processo de busca, uma lista de movimentos proibidos (T), chamada de lista Tabu, é mantida durante todo o processo. Para iniciar o

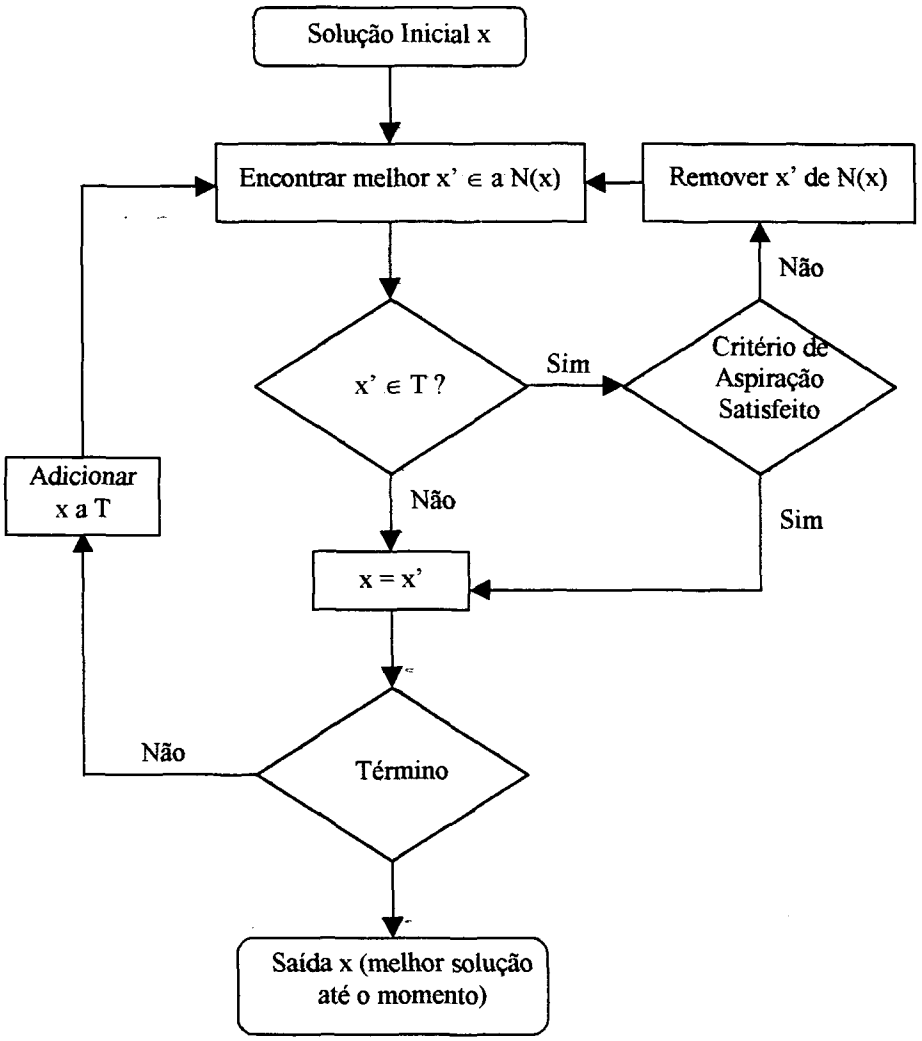

Figura 11 – Estrutura Básica do Método de Busca Tabu (KRISHNAMACHARI, 1999)

processo, seleciona-se uma solução aleatória (x). Uma busca local é então realizada, procurando todas as soluções vizinhas, $N(x)$, a solução inicial. A partir dessas soluções vizinhas, seleciona-se a melhor solução (x'), não sendo necessário que a mesma seja melhor que a solução inicial. A solução inicial é então movida para a melhor solução vizinha, adicionando-se a lista de restrições a nova solução. A partir dessa nova solução, realiza-se novamente uma busca local e novamente a melhor solução vizinha é selecionada como candidata para o próximo movimento.

Contudo, para evitar que um movimento reverso seja realizado, verificam-se os movimentos das iterações anteriores armazenados na lista de restrições. Se o movimento não se encontra na lista Tabu ou se satisfeito o critério de aspiração, aceita-se o movimento, caso contrário testa-se a próxima melhor solução. Este teste é realizado até encontrar uma solução vizinha que não se encontre na lista Tabu. Todo este processo é repetido até que um critério de término seja satisfeito.

4.3. Listas Tabu em Algoritmos Genéticos

KURAHASHI e TERANO (2000), propõe um Algoritmo Genético utilizando-se de múltiplas listas Tabu, auxiliando o algoritmo a alcançar a solução que envolva problemas multimodais ou com mais de uma função objetivo a otimizar.

Uma idéia geral do algoritmo proposto pode ser visualizada na Figura 12. O algoritmo é composto por duas listas de restrições: Lista Longa, com tamanho m e Lista Curta com tamanho n. Estes tamanhos das listas podem ser ajustadas de acordo com o problema. Elas possuem os objetivos de: armazenar os melhores indivíduos das gerações anteriores (regras), manter o elitismo, manter a diversidade da população e evitar a convergência em um ponto ótimo local.

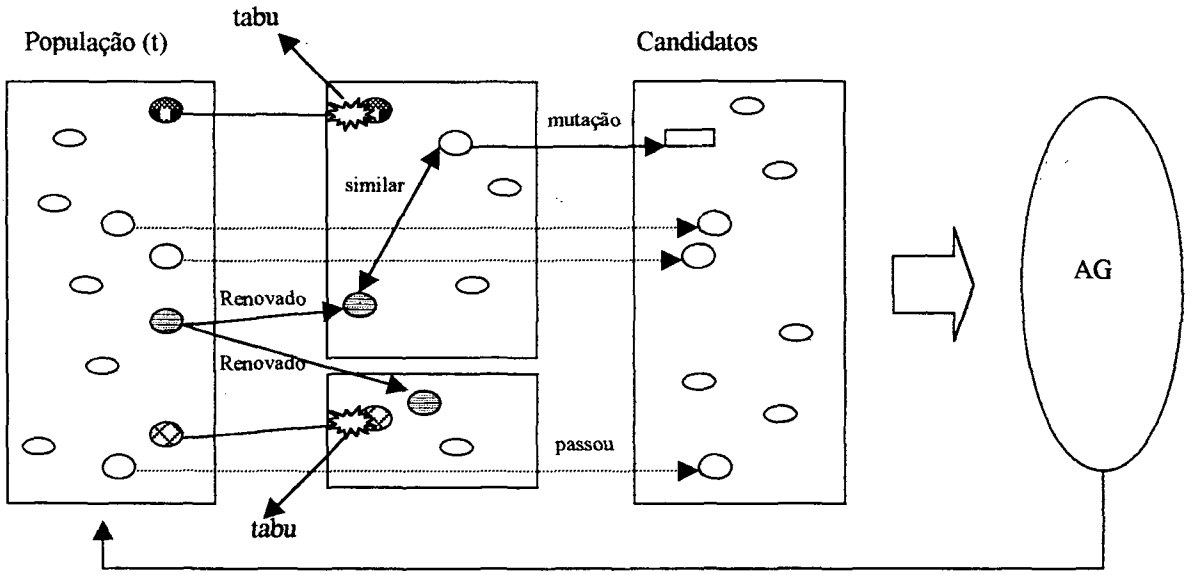

Figura 12 – Ilustração do Algoritmo *Tabu* – AG (KURAHASHI; TERANO, 2000)

A dinâmica dos algoritmos baseia-se na idéia de que ao final de cada geração, o melhor indivíduo será armazenado em ambas as listas. Quando se inicia a próxima geração e novos indivíduos são selecionados para o cruzamento, as listas de restrições não deixam que indivíduos similares presentes nas mesmas sejam selecionados. É importante notar que elas serão aplicadas apenas a um dos "pais" a gerar os "descedentes". Outro fator a ser levado em conta é que alguma função de medida de distância deverá ser adotada, seja ela no espaço fenotípico ou genotípico.

Na Lista Curta, com tamanho n, serão armazenados apenas os indivíduos das iterações mais recentes. Quando a lista é preenchida completamente, para que um novo indivíduo seja adicionado a mesma, o indivíduo mais antigo deverá ser retirado. Os indivíduos pertencentes a esta lista podem ter o mesmo genótipo.

Quanto à Lista Longa, serão armazenados indivíduos de todas as gerações anteriores. Os indivíduos presentes na lista não poderão ter genótipo idêntico ou similar. Caso surja um indivíduo com um genótipo similar a ser adicionado na Lista Longa, este somente será adicionado, se possuir um valor de função de aptidão superior e mediante a retirada do outro indivíduo. Este indivíduo retirado da lista, sofrerá mutação e será recolocado na população a fim de participar das próximas gerações.

Assim, as soluções serão gradualmente armazenadas na Lista Longa, ou seja, até no máximo *m* soluções. Ao final do processo, o conjunto solução, será formado por indivíduos presentes na Listas Longas.

Testes de otimizações de funções realizados por este algoritmos foram feitos por KURAHASHI e TERANO (2000). Algumas de suas conclusões parciais relatam que com a adoção desta estratégia, o Algoritmo Genético utilizando listas Tabu, conseguiu cobrir uma área maior do espaço de busca, se comparado a um Algoritmo Genético puro. Além disso, este método apresentou-se poderoso e robusto em problemas com muitas classes para classificação.

A abordagem desta dissertação fará justamente a utilização deste último método para manter a diversidade durante o processo de busca. Como comentado anteriormente, considerou-se este método interessante e ainda não explorado em processos de classificação com Algoritmos Genéticos, já que KURAHASHI e TERANO (2000), não investigaram seus algoritmos neste sentido.

Espera-se, com a implementação deste método e análise dos resultados, que o presente trabalho contribua com a busca de novos caminhos para um melhor desempenho de um Algoritmo Genético.

Detalhes desta implementação, serão abrodados no Capítulo 5. Onde preliminarmente comentaremos o processo de geração de um classificador e a heurística para a junção de regras, associadas ao uso de um Algoritmo Genético.

5. Considerações na implementação do Algoritmo Genético

A utilização de Algoritmos Genéticos no processo de geração de um classificador, envolvem algumas considerações a serem adotadas em relação a um Algoritmo Genético Padrão. Assim, procura-se neste capítulo, descrever de forma mais detalhada algumas das abordagens utilizadas para a etapa de geração de um classificador, bem como as heurísticas utilizadas no processo de busca para o algoritmo implementado.

5.1. Classificador

Quando se pretende implementar um classificador baseado em um Algoritmo Genético, pode-se adotar uma divisão deste processo em três etapas: estratégias de decomposição, tamanho da população e criação do conjunto de regras (HASSE; POZO, 2000) . A seguir, apresentamos mais detalhadamente cada uma destas etapas.

5.1.1. Estratégias de Decomposição

Antes de realizar o processo de busca em qualquer Algoritmo Genético a ser implementado, o primeiro passo a ser definido é o tratamento dado às classes pertencentes ao conjunto de dados .

Duas abordagens podem ser utilizadas: uma única população de regras para todas as classes de dados e conjunto de regras para cada classe do conjunto de dados a ser analisado, trabalhando com Algoritmos Genéticos independentes. Assim, nesta segunda abordagem, os dados a serem analisados são simplesmente divididos entre exemplos pertencentes à classe a ser obtido o classificador (positivos) ou exemplos pertencentes às demais classes (negativos).

Segundo HASSE (2000), implementações de um classificador que utilizavam uma única população de regras para todas as classes, tiveram dificuldade para gerar regras úteis, ou seja, regras capazes de cobrir um ou mais exemplos de treinamento. Em todos os testes realizados para diferentes conjuntos de dados, foram necessários vários cruzamentos até se alcançar uma regra válida, capaz de classificar, pelo menos, um exemplo de treinamento. Nas conclusões do referido autor, este fato ocorreu devido a maioria dos indivíduos conter

pares de atributos com valores que não ocorriam simultaneamente em nenhum exemplo de treinamento.

Em contrapartida, a utilização de populações divididas por classes, tornou a evolução das regras muito mais rápida pois, apenas valores válidos de atributos foram utilizados, fazendo com que apenas alguns cruzamentos raros, resultassem em regras que não classificassem nenhum exemplo de treinamento.

Algumas outras vantagens podem ser alcançadas com a utilização de separações das populações por classes, as quais são apresentadas a seguir:

- O espaço de busca pode ser bastante reduzido pelo uso apenas de valores "potencialmente úteis" dos dados de treinamento (NODA; FREITAS; LOPES, 1999). Quando a busca é dirigida apenas a uma classe, é necessário analisar apenas valores de atributos que apareçam em exemplos positivos daquela classe. Esta condição pode ser implementada na criação da população inicial onde, segundo HASSE (2000), esta abordagem reduz drasticamente a quantidade de regras da população, incapaz de classificar exemplos positivos.

- A implementação do algoritmo e o seu controle torna-se mais simples. Além disso, através dessa abordagem, consegue-se reduzir o tempo de processamento pois, ao gerar uma população inicial, tem-se a garantia que uma regra selecionada irá classificar, pelo menos, um exemplo positivo.

- A paralelização e distribuição de processamento torna-se fácil, uma vez que cada Algoritmo Genético pode ser processado de maneira independente e utilizando diferentes processadores.

Assim, baseando-se nestas vantagens descritas, adotou-se no presente trabalho, a implementação de populações sub-divididas por classes.

5.1.2. Tamanho da População

Em uma segunda etapa, temos a definição da população. Através da abordagem de populações sub-divididas por classes, torna-se possível adotar dois critérios para definir o tamanho da população inicial a ser utilizada, conforme segue:

- **Distribuição Proporcional por Classes**

  Neste sistema, o número de exemplos pertencentes a cada classe é utilizado como base para criar populações de regras com aproximadamente a mesma proporção.

- **Distribuição igual por Classes**

  O número de regras da população de cada sub-processo é o mesmo para cada classe. A adoção deste sistema de distribuição, auxilia na previsão dos exemplos que pertencem às classes menos freqüentes.

Definido o critério a ser adotado, inicia-se o processo de busca, envolvendo os conceitos de Algoritmos Genéticos que por sua importância, serão tratados separadamente na seção 5.2.

5.1.3. Geração das Regras

Efetuado o processo de busca e, tendo obtido um conjunto de soluções, o classificador deve então organizar as regras de forma adequada, visando obter uma precisão satisfatória. A forma de agrupamento das regras pode ser feita em conjuntos ordenados ou não ordenados.

Segundo CLARK e BOSWELL (1991), ao se utilizar conjuntos de regras não-ordenadas, não há uma ordem pré-estabelecida de utilização de qual regra será utilizada primeiro para classificar uma instância. Todas as regras serão utilizadas simultaneamente. Assim, caso mais de uma regra classifique uma instância e, as mesmas indiquem classes diferentes, a classe escolhida dependerá de um critério de soma dos

valores ou pesos atribuídos a cada regra. Este tipo de abordagem pode dificultar a compreensão quando muitas regras classificam sub-conjuntos comuns de exemplos.

Em relação à abordagem de conjuntos de regras ordenadas, todas as regras são organizadas em uma determinada ordem. Desta forma, para classificar uma nova instância, será seguida esta ordem pré-determinada, testando regra a regra, até encontrar uma que classifique o exemplo, finalizando o processo de classificação. Quando nenhuma das regras atende ao exemplo analisado, é adotada uma classe padrão para a classificação da instância.

Um pseudo-código para a geração de um conjunto de regras ordenadas é apresentado na Figura 13 (CLARK; BOSWELL, 1991). Inicialmente cria-se um conjunto vazio de regras (classificador). Enquanto o conjunto de dados a analisar (exemplos) não estiver vazio, haverá a verificação da existência de alguma regra possível a ser adicionada no conjunto de regras. A melhor regra que classifique a classe mais freqüente de exemplos será selecionada. Se esta regra escolhida melhorar o desempenho do classificador, segundo determinada métrica, a mesma será mantida, removendo-se todos os exemplos cobertos por esta regra, caso contrário, a regra selecionada será descartada.

No caso de não haver mais nenhuma regra a ser adicionada, deverá se proceder a verificação da classe dos exemplos mais freqüentes, para definição de qual será a classe padrão, removendo todos os exemplos remanescentes.

```
Cria-se um conjunto vazio de Regras
Enquanto conjunto de exemplos analisados diferente de vazio, faça
   Se o conjunto de regras a analisar não estiver vazio então
      Retirar a melhor regra que cobre a classe mais freqüente dos exemplos
      Se o classificador melhora com a adição desta nova regra então
         Remover todos os exemplos cobertos pela regra.
      Senão
         Eliminar regra
   Senão
      Definir a classe mais freqüente de exemplos como regra padrão
      e remover todos os exemplos de dados analisados
```

Figura 13 – Pseudo-código para a geração de regras ordenadas (CLARK; BOSWELL, 1991)

Assim, ao utilizar este algoritmo, torna-se necessário adotar uma métrica que avalie adequadamente a melhora de desempenho do classificador com a adição de uma regra. Os principais conceitos envolvidos para análise deste tipo de métrica são:

- **Verdadeiros Positivos (VP)**

  Número de exemplos positivos corretamente classificados como positivos pelo classificador.

- **Verdadeiros Negativos (VN)**

  Número de exemplos negativos corretamente classificados como negativos.

- **Falsos Positivos (FP)**

  Número de exemplos negativos que foram incorretamente considerados pelo classificador como positivos.

- **Falsos Negativos (FN)**

  Número de exemplos positivos incorretamente classificados como negativos.

Existem diferentes critérios para medir a precisão de um classificador. Uma métrica geralmente utilizada é o percentual de acertos (Equação 1), dado pelo número de exemplos classificados corretamente pelo número total de instâncias testadas (NODA; FREITAS; LOPES, 1999).

$$\text{Percentual de acertos} = (\mathbf{VP})/(\mathbf{VP+VN+FP+FN}) \quad \text{(Equação 1)}$$

Além da métrica de avaliação, outro ponto em questão refere-se ao número de regras a serem geradas. O algoritmo acima, se considerado sem nenhuma heurística associada a ele, deverá gerar o maior número possível de regras, desde que cada regra adicionada melhore a precisão do classificador.

O uso de outras heurísticas, ou métricas, pode melhorar a precisão do classificador. Entretanto, a parte mais importante para a eficiência do algoritmo refere-se ao processo de busca de regras.

5.2. O Algoritmo Genético e Listas Tabu

Conforme visto anteriormente, um Algoritmo Genético puro, apresenta a forma esquematizada na Figura 2. Porém, visando a melhora do desempenho do mesmo, muitas vezes é necessário a utilização de meta-heurísticas adicionais, além da realização de certas modificações em relação aos operadores do algoritmo. Assim, neste capítulo são discutidos os principais tópicos relacionados a essas especializações.

5.2.1. Sistema de representação das regras

O Algoritmo Genético implementado no presente trabalho, adotou como sistema de indução de regras a abordagem de Michigan, onde cada indivíduo da população corresponde a uma regra simples. Desta forma, cada uma destas regras foi representada com o seguinte formato:

$$R_1 \wedge R_2 \wedge \ldots R_i \Rightarrow C \qquad , i = 1, \ldots n$$

Sendo:

- n, número de atributos;.
- R, possíveis cláusulas de restrição para o valor do atributo i;
- C, classe correspondente a regra específica.

Segundo HASSE (2000), esta forma de codificação, torna possível que qualquer restrição sobre qualquer atributo, possa ser nula; ou seja, pode haver uma ou mais restrições em uma regra que considerem válidos quaisquer valores pertencentes ao atributo sobre o qual a restrição se aplica.

5.2.2. População Inicial

Duas abordagens foram adotadas na implementação para a geração da população inicial. Na primeira abordagem, chamada de semente da população, exemplos são selecionados aleatoriamente a partir do conjunto de dados de treinamento. Desta forma, cada indivíduo selecionado, é utilizado como semente para a geração de uma regra. A partir desta regra, algumas das restrições poderão ser modificadas, podendo ter seus valores alterados para cláusulas vazias, conforme um índice de probabilidade a ser especificado.

Na segunda abordagem, semente aleatória, exemplos são criados aleatoriamente pelo sistema. O algoritmo verifica quais restrições são necessárias a formação de um indivíduo e, sorteia aleatoriamente cada um dos possíveis valores a serem assumidos pelos diferentes atributos. Da mesma forma como a abordagem anterior, é permitida a ocorrência de cláusulas vazias em uma regra.

Segundo HASSE (2000), o coeficiente de criação de restrições vazias pode afetar de forma significativa o comportamento do algoritmo. Quando utilizado um coeficiente muito alto, serão criadas regras com muitas restrições vazias e a busca ocorrerá de regras gerais para regras específicas. Em contrapartida, utilizando-se valores baixos de coeficiente, a busca ocorrerá em sentido inverso, iniciando com regras específicas, as quais serão gradativamente substituídas, à medida que regras mais gerais entrem na população.

Finalizado o processo de geração da população inicial, inicia-se uma segunda etapa, o cálculo da função *fitness* de todos os indivíduos da população.

5.2.3. Função *Fitness*

A função *fitness* tem um papel crucial em um Algoritmo Genético. A partir dela, o algoritmo poderá avaliar suas regras e continuar o processo de evolução das mesmas. Diferentes variações de funções de avaliação vêm sendo implementadas e testadas, porém muitas delas são dependentes do domínio de dados que se deseja avaliar. Uma função *fitness* implementada com bons resultados, em diferentes domínios, é a função de Laplace (NIBLETT, 1987). Alguns exemplos de aplicação podem ser encontrados em CLARK e NIBLETT (1989) e LOPES et al. (2000).

Assim, a implementação atual adotou a utilização da função de Laplace, a qual é descrita na Equação 2.

$$\text{Função } \textit{fitness} = (VP+1)/(VP+FP+K) \qquad \text{(Equação 2)}$$

Sendo:

- K, o número de classes no domínio.
- VP, número de verdadeiro positivos.
- FP, número de falsos positivos.

5.2.4. Operador de Seleção

Na implementação realizada, apenas o método de Seleção de Torneio foi implementado. A razão principal se deve ao fato de o mesmo apresentar uma evolução mais estável e possuir um custo computacional menor se comparado a outros métodos (MITCHELL, 1997). Futuras implementações poderão testar outros métodos.

5.2.5. *Crossover*

Para o processo de *crossover* foi utilizado o cruzamento uniforme. A partir de uma máscara de entrada, os genes dos "pais" selecionados são recombinados, onde essa recombinação é realizada característica por característica. A ocorrência de *crossover* está vinculada à taxa de probabilidade fixada pelo usuário.

Além disso, no sentido de evitar o "incesto", cruzamento de indivíduos idênticos, uma restrição impede a ocorrência desse fato, exceto se o algoritmo não conseguir encontrar um indivíduo diferente em mais de dez tentativas. O objetivo dessa restrição, foi evitar um dispêndio computacional desnecessário para gerar indivíduos idênticos.

5.2.6. Mutação

Conforme comentado anteriormente, o operador de mutação visa manter a diversidade da população. Assim, mediante uma probabilidade (em geral 2%), o operador simplesmente escolhe uma das cláusulas da regra e altera aleatoriamente este valor, dentre os valores possíveis para esta característica. Além disso, está cláusula também pode ser alterada para uma cláusula vazia, visando além de aumentar a diversificação, também provocar a generalização das regras. Porém, este fato apenas ocorrerá mediante a probabilidade de cláusulas vazias especificadas pelo usuário e quando tiver todas as suas cláusulas vazias.

5.2.7. Listas Tabu

A utilização de listas Tabu no algoritmo implementado, teve por objetivo procurar manter uma maior diversidade da população durante a evolução do Algoritmo Genético. Desta forma, pode obtener-se ao final da execução um conjunto de regras potenciais para integrar o classificador. Além disso, o algoritmo implementado permite optar pela utilização ou não das listas Tabu. Isto permite comparar a efetividade desta abordagem.

A estratégia do uso de listas Tabu adotada foi a mesma utilizada na implementação por KURAHASHI e TERANO (2000), considerando-se o problema de busca para funções multimodais.

O algoritmo trabalha com duas listas Tabu, lista Longa e lista Curta, para cada uma das classes a ser obtido o classificador. Inicialmente vazias, ao término de cada geração, o melhor indivíduo de cada classe é adicionado às suas respectivas listas Tabu. Algumas considerações importantes devem ser levadas em conta no processo de adição de uma restrição:

- **Listas longas**

  Nessas listas não é permitida a existência de restrições ou indivíduos iguais ou similares (definido por uma medida de distância). Caso um indivíduo similar a outro pertencente à lista Tabu tente ser adicionado a mesma, prevalecerá na lista o indivíduo com maior

valor de *fitness*. No caso da adição de indivíduos diferentes aos pertencentes a Lista Longa, estes somente serão adicionados no caso em que a lista não esteja totalmente preenchida (tamanho da lista).

- **Listas curtas**

  Nestas listas poderão ser adicionados indivíduos iguais ou similares de gerações anteriores. Cada uma dessas restrições será mantida por um certo período de gerações, as quais estarão vinculadas ao tamanho da lista. Se um novo indivíduo deve ser adicionado à lista curta e, a mesma esteja preenchida, então o indivíduo mais antigo deverá ser removido da mesma.

A partir da criação e incremento dessas listas Tabu, o papel principal das mesmas será o de inibir a seleção de, somente um dos dois indivíduos envolvidos em um processo de cruzamento, levando-se novamente em conta uma medida de distância.

Segundo KURAHASHI e TERANO (2000), esta medida de distância pode ser realizada tanto no espaço genotípico como fenotípico. Como no presente trabalho, a implementação realizada de busca foi baseada no espaço fenotípico, a medida de distância adotada para função de similaridade representa a diferença dos fenótipos de cada gene do indivíduo.

Assim, baseando-se nas considerações comentadas anteriormente, o algoritmo foi implementado de forma que parâmetros como tamanho das listas e valor da distância fossem fornecidos pelo usuário, podendo este realizar um ajuste mais refinado, de acordo com o domínio a ser estudado.

6. O Algoritmo de Geração de um Classificador

Neste capítulo se apresenta se aspectos genéricos referentes a implementação do algoritmo restrito por listas Tabu, incluem-se também aspectos de validação deste. Foram realizados dois experimentos: o primeiro visando analisar o efeito das listas tabu no processo de busca e o segundo, comparando o mesmo frente a outros algoritmos.

6.1. Visão Geral

O algoritmo implementado consiste em um classificador baseado na indução de regras através do Algoritmo Genético descrito anteriormente. Todos os parâmetros do algoritmo podem ser definidos através de uma interface gráfica simples, mostrada na Figura 14.

A linguagem de programação escolhida para a implementação desse algoritmo foi

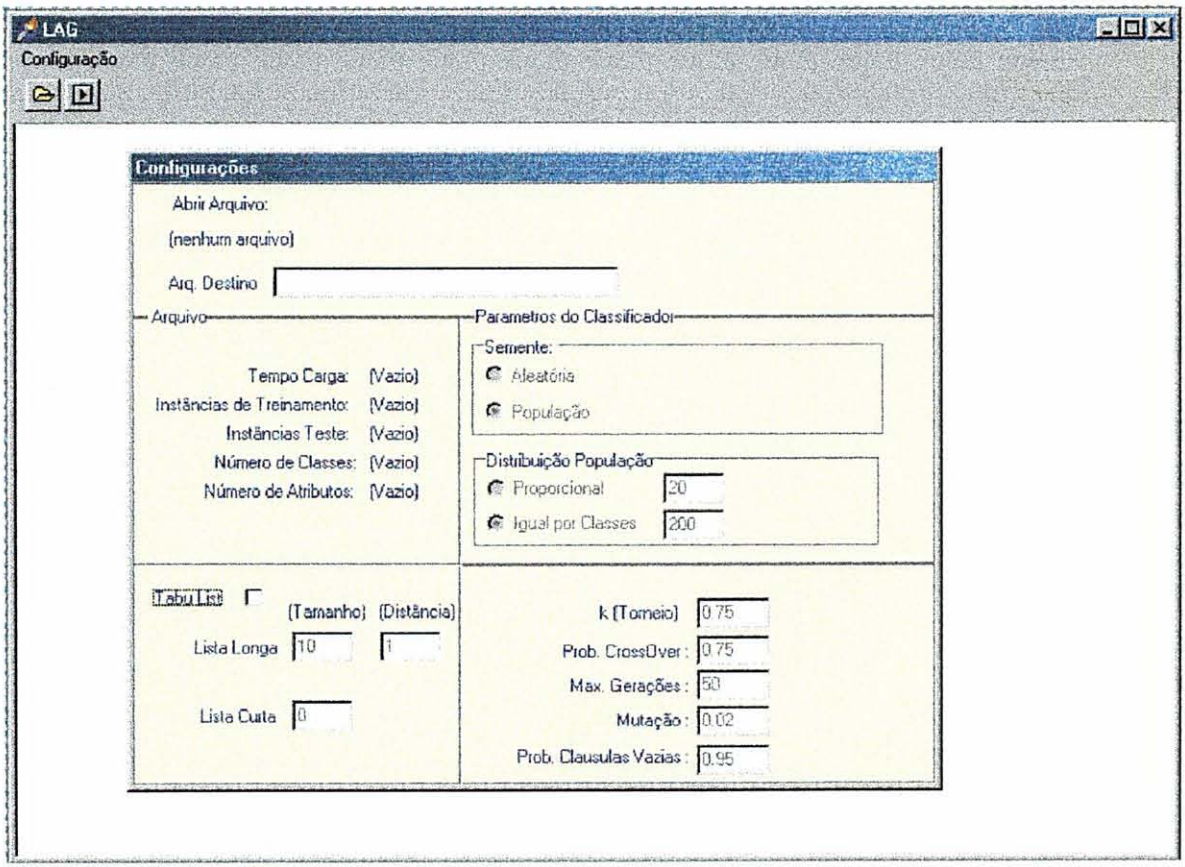

Figura 14 – Interface gráfica do algoritmo implementada e seus parâmetros de configuração

Delphi®. A principal razão para a utilização dessa linguagem na implementação do algoritmo foi a facilidade de uma possível união entre os módulos principais dos algoritmos com a ferramenta proposta por HASSE (2000).

Além disso, a linguagem permite maior simplicidade e clareza, tornando o código bastante modular devido à organização das classes dos objetos. Seu ambiente gráfico também permite o uso de recursos visuais, criando um ambiente fácil de trabalhar, uma vez que melhora a comunicação entre homem e máquina.

O primeiro passo para iniciar a ferramenta é a abertura do conjunto de dados que originará o classificador. Adotou-se trabalhar com o formato de dados utilizados pela ferramenta C4.5 (QUINLAN, 1993), devido o fato da maioria das bases de dados disponíveis para experimentos realizados em pesquisa se encontrar neste formato, o qual baseia-se na utilização de três arquivos, descritos a seguir:

- **Extensão *names***

Contém as classes dos dados, nomes, ordem e tipos dos atributos.

- **Extensão *data***

Contém dados de treinamento.

- **Extensão *test* (opcional)**

Contém dados de teste para medir a precisão do classificador em dados novos ou desconhecidos.

O próximo passo, após a escolha do conjunto de dados, é a definição de todos os parâmetros a serem utilizados pelo algoritmo. Eles envolvem desde o critério de segmentação das classes, tamanho da população, número de gerações, uso ou não de listas Tabu , assim como parâmetros tradicionais de um Algoritmo Genético. Nenhuma tentativa foi realizada no sentido do próprio sistema definir automaticamente quais os melhores parâmetros a serem utilizados; todos eles obrigatoriamente deverão ser definidos manualmente pelo usuário.

Delphi® é marca registrada da Interprise

Durante a execução do programa, uma janela é exibida (Figura 15), indicando a etapa atual de execução do algoritmo, bem como o tempo consumido por cada uma das etapas. Ao final do processo de execução, uma nova janela é exibida, onde se encontram informações básicas do arquivo; percentuais de avaliação dos conjuntos de dados de treinamento e teste; indicação do valor da *fitness* mínima, máxima e média da população, em cada geração (Figura 16); o conjunto de regras geradas (Figura 17) e; a distribuição de acertos de classificação de cada regra pelo conjunto de dados (Figura 18).

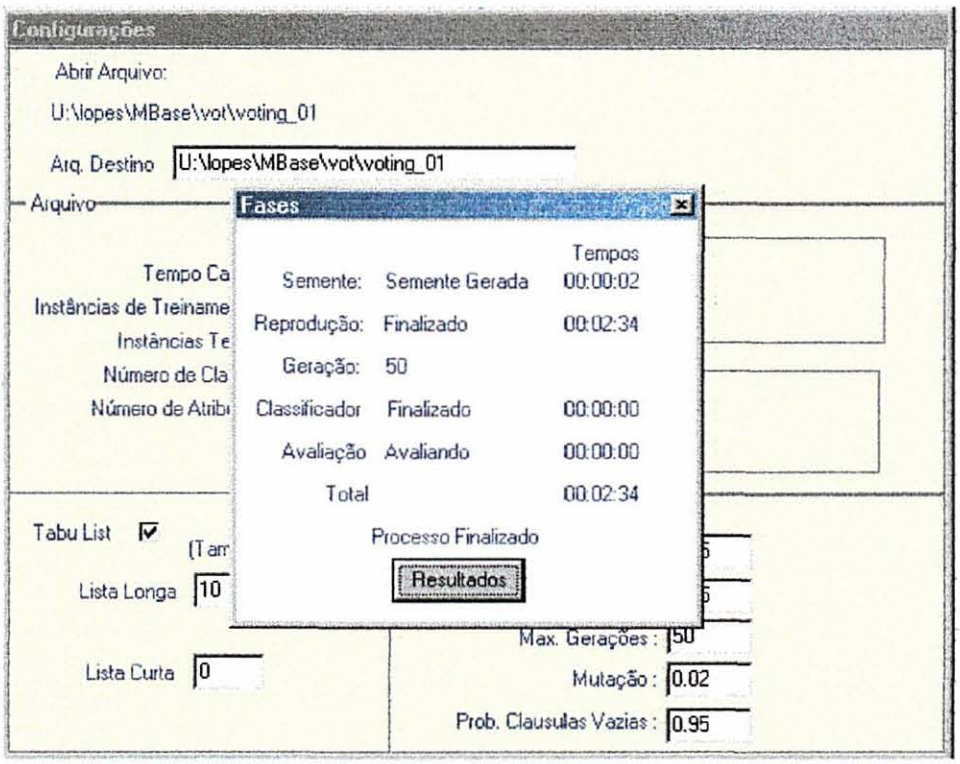

Figura 15 - Janela de acompanhamento de execução do algoritmo

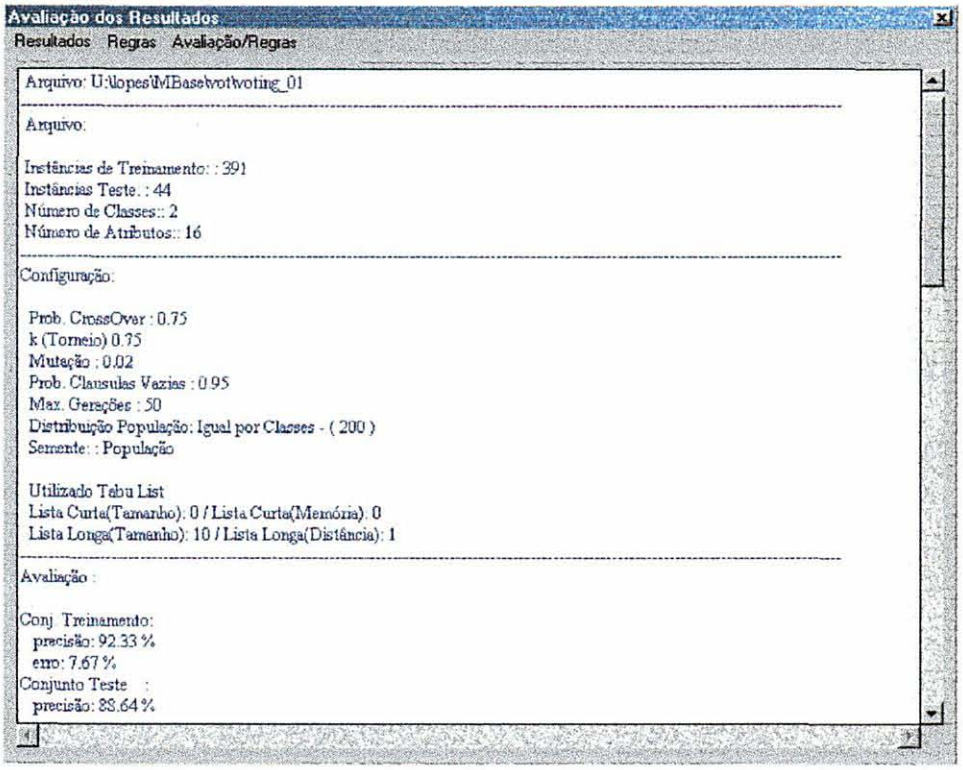

Figura 16 - Janela com os resultados básicos do classificador

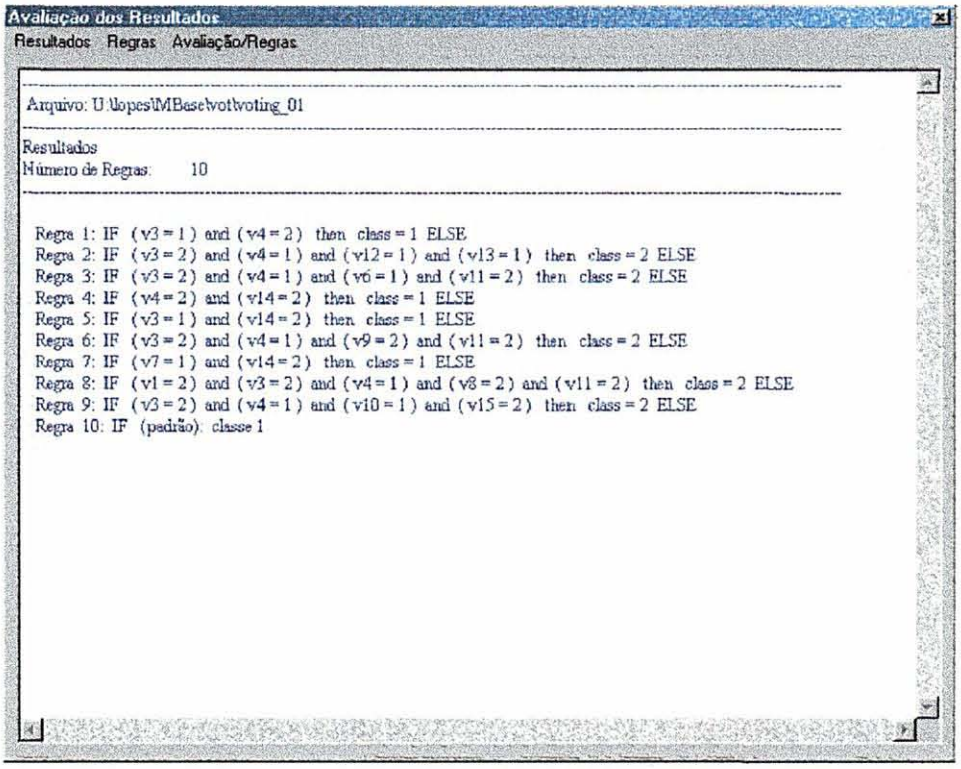

Figura 17 - Janela com o conjunto de regras geradas

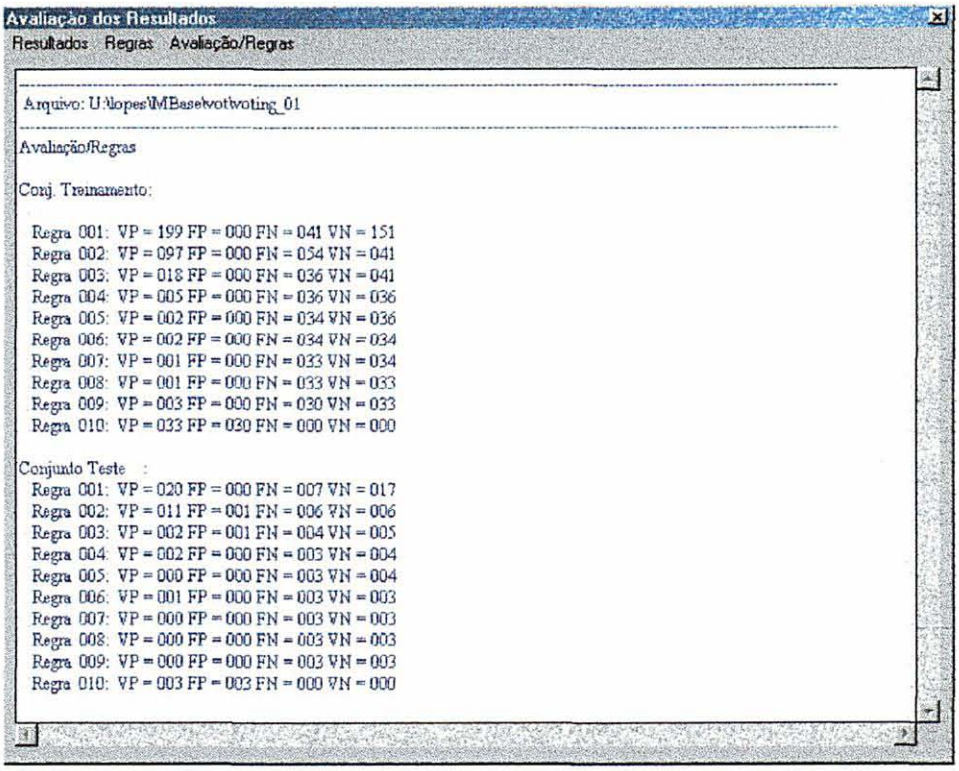

Figura 18 - Janela com a distribuição de acertos de classificação de cada regra

6.2. Verificação do efeito das Listas Tabu no processo de busca

Para melhor entender e verificar o efeito da utilização das Listas Tabu no Algoritmo implementado, foram realizados experimentos com a base de dados *vot* (descrita mais datalhadamente na seção 6.3), obtendo a partir de diferentes gerações, os conjuntos de regras presentes nas populações, Listas Longas e Listas Curtas. A seguir apresenta-se os resultados obtidos na 40ª geração da classe 2, dessa base de dados, na Figura 19 é representado o conjunto de regras da Lista Curta; na Figura 20, observa-se os conjunto de regras da Lista Longa e no Anexo1 encontra-se o conjunto completo de regras da população.

Fitness	Conjunto de Regras
0.989	Regra 1: IF (v3 = 2) e (v4 = 1) e (v11 = 2) e (v14 = 1) e (v15 = 2) then class = 2
0.989	Regra 2: IF (v3 = 2) e (v4 = 1) e (v11 = 2) e (v14 = 1) e (v15 = 2) then class = 2

Figura 19 - Conjunto de Regras da Lista Curta

Fitness	Conjunto de Regras
0.983	IF (v3 = 2) e (v4 = 1) e (v10 = 1) e (v12 = 1) then class = 2
0.980	IF (v1 = 2) e (v3 = 2) e (v4 = 1) e (v14 = 1) then class = 2
0.989	IF (v3 = 2) e (v4 = 1) e (v9 = 2) e (v11 = 2) then class = 2
0.990	IF (v3 = 2) e (v4 = 1) e (v11 = 2) then class = 2
0.989	IF (v1 = 2) e (v3 = 2) e (v4 = 1) e (v11 = 2) then class = 2
0.989	IF (v1 = 2) e (v3 = 2) e (v4 = 1) e (v5 = 1) e (v13 = 1) then class = 2
0.987	IF (v3 = 2) e (v4 = 1) e (v7 = 2) e (v9 = 2) e (v12 = 1) e (v13 = 1) e (v14 = 1) then class = 2
0.989	IF (v3 = 2) e (v4 = 1) e (v6 = 1) e (v11 = 2) e (v14 = 1) then class = 2
0.987	IF (v3 = 2) e (v4 = 1) e (v6 = 1) e (v7 = 2) e (v12 = 1) e (v13 = 1) e (v14 = 1) then class = 2
0.987	IF (v1 = 2) e (v3 = 2) e (v4 = 1) e (v12 = 1) e (v13 = 1) e (v14 = 1) e (v15 = 2) then class = 2

Figura 20 - Conjunto de Regras da Lista Longa

6.2.1. Análise do Efeito das Listas Tabu

Através da Figura 19, observa-se que na Lista Curta existem duas regras idênticas, mostrando que estes dois indivíduos representam as melhores regras obtidas, nas duas últimas gerações da classe 2 . Ñão obstante, analisando os conjuntos de regras da população (no item Anexos), pode-se observar que a mesma apresenta uma unica ocorrência do indivíduo presente na Lista Curta.

Além disso, a partir da Figura 20, pode-se verificar que na Lista Longa não existe a presença do indivíduo da Lista Curta. Isto justifica-se pela regra de adição da Lista Longa. Estes dois indivíduos não deveriam ser adicionados à mesma, já que possuem indivíduos similares na Lista Longa com maior fitness (de acordo com a medida de distância utilizada).

6.3. Comparação do Algoritmo implementado frente a outros classificadores

No sentido de poder analisar o desempenho do algoritmo implementado, foram realizados experimentos semelhantes aos propostos por LIM, LOH e SHIH (1999) e HASSE (2000). Através de cinco bases de dados, avaliou-se a precisão do classificador, o tempo de execução do algoritmo, o número de regras obtidas, a utilização de populações menores e, o comportamento do algoritmo frente a presença de ruídos.

A forma de avaliação de precisão do classificador utilizada foi a medição da taxa percentual de classificações incorretas em conjuntos de dados teste, obtendo o percentual total de erros, ou taxa de erros.

Além disso, de acordo com o tamanho das bases, adotou-se a seguinte metodologia para a realização dos experimentos:

- **Grande Volume de Dados**

Considera-se volume alto de dados, quando o conjunto de dados é bem maior que 1000 casos, e o arquivo teste com pelo menos 1000 casos para avaliar o classificador gerado pelo conjunto de treinamento.

- **Pequeno Volume de Dados**

Quando trabalha-se com pequeno volume de dados, utiliza-se um procedimento chamado de Cross-Validation (LIM; LOH; SHIH, 1999). Neste, os dados originais são divididos em 10 sub-conjuntos distintos, sem exemplos em comum. Cada um dos sub-conjuntos contém aproximadamente a mesma proporção de exemplos de cada classe do conjunto original. Para cada sub-conjunto, um classificador é criado usando-se como dados de treinamento os outros 9 sub-conjuntos. O classificador gerado é então avaliado, utilizando-se como teste o sub-conjunto não incluído nos dados de treinamento, resultando em uma taxa de erro parcial. A taxa de erro final é obtida pela média de todas as 10 estimativas dos sub-conjuntos.

Para verificar a eficácia das listas Tabu em manter a diversidade das populações, foram realizados testes com diferentes tamanhos para uma mesma população, comparados à variação da taxa de erro de classificação.

Na análise de desempenho do algoritmo proposto, os resultados de taxa de erro de classificação, tempo de execução e número de regras, foram comparados aos resultados de outros 34 algoritmos, encontrados em LIM, LOH e SHIH (1999) e HASSE (2000).

6.3.1. Dados utilizados nos experimentos

Os conjuntos de dados analisados pertenceram a 5 domínios diferentes (STATLIB, 1994; BLAKE; MERZ, 1998). Para aumentar a quantidade de dados analisados e também testar os algoritmos quanto à presença de ruído para cada domínio, foi criado um outro conjunto com todos os dados do conjunto original, cada um acrescido de alguns atributos semelhantes mas com valores aleatórios. Os conjuntos de dados acrescidos de ruído foram diferenciados pela presença do sinal "+" após o nome abreviado, segundo recomendação de LIM, LOH e SHIH (1999).

Uma breve descrição dos dados e suas características se encontram na Tabela 3.

Tabela 3 - Conjuntos de dados analisados (LIM; LOH; SHIH, 1999)

Nome	Tam.	Classes	Atrib. Contínuos	Atrib. Discretos	Breve descrição do domínio
led	2000	10		7	Identificação de dígitos pelos leds acesos
led+	2000	10		24	(Domínio criado artificialmente)
bld	345	2	6		Diagnóstico de doenças de fígado em homens
bld+	345	2	15		via testes de sangue e consumo de álcool
pid	532	2	7		Diagnóstico de diabetes de pacientes da Índia
pid+	532	2	15		Diagnóstico pelo estado psicológico + testes
smo	1855	3	3	5	Previsão da atitude de pessoas diante de
smo+	1855	3	10	5	restrições a fumantes no local de trabalho.
vot	435	2		16	Classificação de um político como republicano
vot+	435	2		30	ou democrata de acordo com suas votações

6.3.2. Algoritmos utilizados como base de comparação

Em LIM, LOH e SHIH (1999), foram realizadas comparações entre diferentes algoritmos, alguns baseados em árvores de decisão, algoritmos estatísticos e redes neurais. Além desses algoritmos, o presente trabalho também comparou os resulrados da implementação proposta por HASSE (2000), baseada em um Algoritmo Genético. A seguir, é apresentada uma breve descrição dos mesmos.

- **Árvores de Decisão**

Os Algoritmos baseados em Árvores de Decisão, utilizados na comparação, foram os seguintes:

CART – Versão do algoritmo CART (BREIMAN et al., 1984), tendo um critério especial de escolha de atributos. Dois métodos de poda foram testados.

Árvore S-Plus – variante do método CART escrito na linguagem S (BECKER; CHAMBERS; WILKS, 1988). É descrita em detalhes em CLARK e PREGIBON (1993).

C4.5 – foi usada a *release* 8 (QUINLAN, 1993; QUINLAN, 1996), com valores padrão, gerando árvores de decisão e usando o programa do pacote para a geração de conjuntos de regras.

FACT – é um algoritmo rápido de classificação, detalhado em LOH e VANICHSETAKUL (1988). Duas variantes foram testadas conforme o critério de escolha dos atributos.

QUEST – este algoritmo é descrito em LOH e SHIH (1997). Quatro variações diferentes foram testadas, conforme o critério de poda e escolha dos atributos. A versão usada foi a 1.7.10.

IND – este algoritmo é detalhado em BUNTINE (1992). A versão usada é a 2.1, com valores padrão. Quatro variações são analisadas.

OC1 – Este algoritmo é detalhado em MURTHY, KASIF e SALZBERG (1994). LMDT – detalhado em BRODLEY e UTGOFF (1995), usando os valores padrão do programa.

CAL5 – criado principalmente para valores numéricos, é detalhado em MULLER e WYSOTZKI (1994) e MULLER e WYSOTZKI (1997). Usou-se uma ferramenta do pacote para encontrar os valores ótimos.

T1 – Árvore de decisão baseada na escolha de apenas 1 atributo (HOLTE, 1993).

- **Algoritmos Estatísticos**

Os Algoritmos Estatísticos utilizados na comparação foram os seguintes:

LDA – análise linear do discriminante. Foi usado o SAS PROC DISCRIM (SAS, 1990) com valores padrão.

QDA – análise quadrática do discriminante. Foi usado o SAS PROC DISCRIM (SAS, 1990), com valores padrão.

NN - Implementação do método do vizinho mais próximo do pacote SAS PROC DISCRIM (SAS, 1990).

LDA – *Polytomous Logistic Regression* (AGRESTI, 1990). Análise logística do discriminante.

FDA – análise flexível do discriminante (HASTIE; BUJA; TIBSHIRANI, 1994). Duas variações de (FRIEDMAN, 1991) foram analisadas.

PDA – trata-se de uma variante do algoritmo LDA mostrada em HASTIE; BUJA e TIBSHIRANI (1995).

MDA – análise mista do discriminante. Implementada usando S-Plus e a biblioteca mda (HASTIE; TIBSHIRANI, 1996).

POL – este é o algoritmo *POLYCLASS*, detalhado em KOOPERBERG; BOSE e STONE (1997). Indicado por LIM, LOH e SHIH (1999) como um dos melhores de todos os algoritmos analisados.

- **Redes Neurais**

  O Algoritmo de Redes Neurais utilizado na comparação foi:

  LVQ – *Learning Vector Quantization*, explicado detalhadamente em KOHONEN (1995).

- **Algoritmos Genéticos**

  O Algoritmo de AGs utilizado na comparação foi:

  ALGen – Algoritmo Genético (HASSE, 2000).

6.3.3. Resultados e Discussão

Para analisar o desempenho do algoritmo implementado, foram realizados testes em cinco bases de dados e comparadas a 34 algoritmos diferentes. O equipamento utilizado para este estudo foi um Pentium II ®, com 256 Mb de memória RAM e sistema operacional Windows NT®. Devido a utilização de plataforma e equipamento diferentes dos utilizados por LIM, LOH e SHIH (1999) e HASSE (2000), não pode-se efetuar uma comparação dos tempos de execução dos algoritmos testados por estes pesquisadores.

A partir das Tabelas 4, 5 e 6, podemos observar os resultados do algoritmo implementado, assim como as análises obtidas por LIM, LOH e SHIH (1999) e HASSE (2000). A taxa média de erro de classificação do algoritmo do presente trabalho foi de 0,2591, frente a 0,2423 e 0,2333 obtidas respectivamente por LIM, LOH e SHIH (1999) e HASSE (2000). Estes valores representam uma diferença em média de 1,6% e 2,5% que, segundo LIM, LOH e SHIH (1999), não é significativamente diferente, pois o valor não ultrapassa a margem de 10% do melhor dos classificadores, com estimativa de erro média de 0,195.

Pentium II® é marca registrada da Intel
Windows NT® é marca registrada da Microsoft

Tabela 4 - Dados obtidos a partir do algoritmo implementado

	Trein.	*Test*	Pop.	Classes	Taxa de Erro	Tempo de CPU (hh:mm:ss)
led	2000	4000	600	10	0,2850	00:33:44
led+	2000	4000	1000	10	0,3165	01:11:13
bld	345	345	300	2	0,3775	00:01:52
bld+	345	345	600	2	0,3475	00:07:02
pid	532	532	1000	2	0,2593	00:09:24
pid+	532	532	1000	2	0,2778	00:18:51
smo	1855	1000	2000	3	0,3008	01:57:33
smo+	1855	1000	2200	3	0,3068	03:14:22
vot	435	435	200	2	0,0582	00:02:26
vot+	435	435	200	2	0,0611	00:04:52
Média					0,2591	

Tabela 5 - Conjuntos de dados analisados por LIM, LOH e SHIH (1999) e HASSE (2000)

Dados	Resultados LIM, LOH e SHIH (1999)						Resultados Hasse (2000)
	Taxa Erro			Tempo (hh:mm:ss)			Taxa Erro
	Mín.	Máx.	Mediana	Mín.	Máx.	Mediana	
led	0,2680	0,8160	0,2750	00:00:01	12:25:14	00:04:17	0,2670
led+	0,2650	0,8130	0,2900				0,2760
bld	0,2790	0,4320	0,3220	00:00:04	01:28:47	00:01:47	0,2750
bld+	0,2860	0,4410	0,3440				0,3130
pid	0,2210	0,3100	0,2370	00:00:07	02:27:44	00:02:34	0,2370
pid+	0,2170	0,3180	0,2460				0,2620
smo	0,3040	0,4500	0,3050	00:00:01	03:45:44	00:01:10	0,3040
smo+	0,3050	0,4500	0,3080				0,3090
vot	0,0364	0,0617	0,0480	00:00:02	45:12:23	00:01:59	0,0435
vot+	0,0412	0,0662	0,0480				0,0460
Média			0.2423				0,2333

Tabela 6 - Posição do algoritmo em função da taxa de erro de classificação para cada um dos conjuntos de dados analisados (LIM; LOH; SHIH, 1999; HASSE, 2000)

	Bases de Dados										Média	Pos
	Led	led+	bld	ld+	pid	pid+	smo	smo+	vot	vot+		
QUO	0,2780	0,2770	0,3890	0,4030	0,2260	0,2300	0,3050	0,3050	0,0412	0,0412	0,2495	16
QU1	0,2870	0,2800	0,3990	0,3760	0,2300	0,2300	0,3050	0,3050	0,0435	0,0435	0,2499	16
QL0	0,2740	0,2860	0,3060	0,3800	0,2250	0,2210	0,3050	0,3050	0,0364	0,0503	0,2389	8
QL1	0,2780	0,2900	0,3200	0,3690	0,2230	0,2210	0,3050	0,3050	0,0503	0,0480	0,2409	10
FTU	0,2850	0,2880	0,4010	0,4020	0,2470	0,2580	0,3050	0,3050	0,0435	0,0435	0,2578	19
FTL	0,2710	0,2660	0,4130	0,4190	0,2210	0,2250	0,3050	0,3050	0,0457	0,0457	0,2516	17
C4T	0,2710	0,2900	0,3080	0,3290	0,2420	0,2520	0,3050	0,4050	0,0480	0,0503	0,2500	16
C4R	0,2750	0,3090	0,2920	0,3200	0,2270	0,2330	0,3530	0,3490	0,0526	0,0457	0,2456	15
IB	0,2760	0,3420	0,3280	0,3400	0,2520	0,2780	0,4240	0,4110	0,0526	0,0554	0,2759	25
IBO	0,2740	0,3630	0,3220	0,3330	0,2580	0,2590	0,3930	0,3870	0,0386	0,0506	0,2678	21
IM	0,2740	0,2900	0,3200	0,3400	0,2330	0,2460	0,3050	0,3080	0,0503	0,0594	0,2426	12
IMO	0,2740	0,2960	0,3120	0,3360	0,2460	0,2650	0,3110	0,3390	0,0367	0,0457	0,2461	15
ICO	0,2790	0,2770	0,3270	0,3270	0,2370	0,2330	0,3190	0,3050	0,0480	0,0548	0,2407	9
IC1	0,2860	0,2740	0,3190	0,3130	0,2390	0,2480	0,3050	0,3050	0,0435	0,0457	0,2378	7
OCU	0,2720	0,2690	0,3500	0,3470	0,2370	0,2560	0,3120	0,3050	0,0435	0,0435	0,2435	13
OCL	0,2750	0,3690	0,2960	0,3280	0,3100	0,3180	0,3170	0,3050	0,0574	0,0651	0,2641	20
0CM	0,2800	0,3290	0,2790	0,3670	0,2470	0,2610	0,3050	0,3050	0,0580	0,0662	0,2497	16
STO	0,2850	0,2800	0,3110	0,3260	0,2370	0,2690	0,3050	0,3050	0,0500	0,0500	0,2418	11
ST1	0,2890	0,2900	0,3260	0,3510	0,2500	0,2440	0,3050	0,3050	0,0432	0,0432	0,2446	14
LMT	0,2840	0,3090	0,3220	0,3620	0,2490	0,2530	0,3500	0,3080	0,0483	0,0529	0,2538	18
CAL	0,3010	0,3740	0,4200	0,3980	0,2260	0,2500	0,3160	0,3100	0,0457	0,0457	0,2686	22
T1	0,8160	0,8130	0,4320	0,4410	0,2500	0,2500	0,3040	0,3170	0,0435	0,0435	0,3710	28
LDA	0,2710	0,2650	0,3260	0,3430	0,2210	0,2230	0,3050	0,3110	0,0458	0,0458	0,2357	5
QDA	0,2730	0,2920	0,4010	0,4350	0,2380	0,2560	0,4540	0,4420	0,0549	0,0617	0,2908	26
NN	0,2940	0,4030	0,3700	0,4230	0,2950	0,3130	0,4100	0,4450	0,0526	0,0577	0,3063	27
LOG	0,2690	0,2660	0,3090	0,3440	0,2300	0,2260	0,3050	0,3080	0,0500	0,0435	0,2351	4
FM1	0,2740	0,2710	0,2890	0,3140	0,2220	0,2240	0,3100	0,3230	0,0455	0,0457	0,2318	2
FM2	0,2680	0,2680	0,2800	0,3200	0,2480	0,2280	0,3070	0,3110	0,0432	0,0480	0,2321	2
PDA	0,2740	0,2670	0,3290	0,3430	0,2260	0,2280	0,3050	0,3110	0,0455	0,0455	0,2374	6
MDA	0,2730	0,2670	0,3200	0,3740	0,2310	0,2280	0,3100	0,3090	0,0617	0,0545	0,2428	12
POL	0,2740	0,2680	0,2860	0,2860	0,2370	0,2170	0,3050	0,3050	0,0523	0,0477	0,2278	1
LVQ	0,3130	0,4130	0,3290	0,3430	0,2430	0,2670	0,3660	0,3580	0,0500	0,0614	0,2743	23
RBF	0,4460	0,4300	0,3300	0,3460	0,2300	0,2400	0,3070	0,3170	0,0523	0,0503	0,2749	24
Hasse (2000)	0,2670	0,2760	0,2750	0,3130	0,2370	0,2620	0,3040	0,3090	0,0435	0,0460	0,2333	3
Algoritmo Proposto	0,2850	0,3165	0,3075	0,3475	0,2593	0,2778	0,3008	0,3068	0,0582	0,0611	0,2521	17

Em termos de classificação por algoritmo, a implementação proposta ocupou a 17ª posição em relação aos 34 algoritmos, tendo no total 22 classificadores com resultados superiores a mesma. Em nenhum dos estudos realizados com o algoritmo implementado, os valores obtidos da taxa de erro foram superiores aos classificadores de pior desempenho (Gráfico 1).

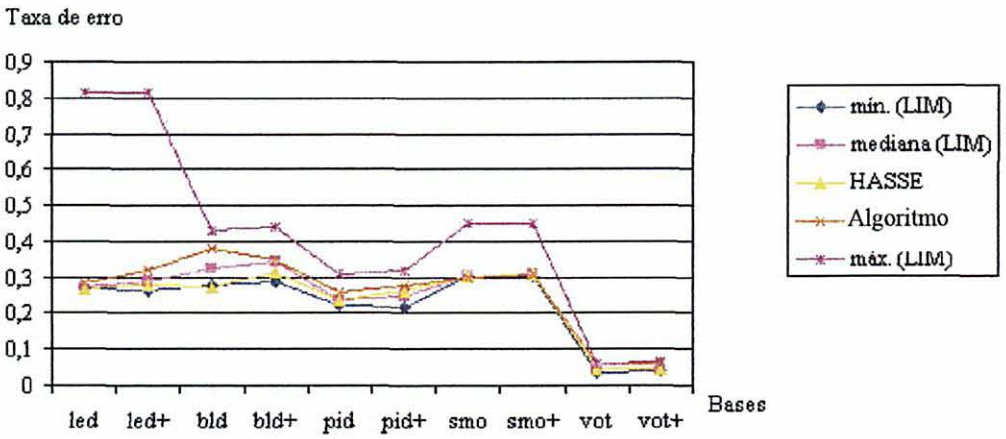

Gráfico 1 - Comparação entre os resultados obtidos pelo Algoritmo proposto e os outros algoritmos (LIM; LOH; SHIH, 1999; HASSE, 2000)

A comparação do algoritmo em relação aos classificadores baseados em regras (Tabela 7), permitiu verificar que, na média, o número de regras geradas foi superior a estes. A principal influência desse resultado ocorreu em função das bases pid, smo e vot. Apesar do Algoritmo implementado poder controlar o número de regras através do tamanho das listas Tabu, percebeu-se ao longo do experimento que, o número de regras era mais afetado pela medida de distância da Lista Longa do que pelo tamanho das Listas.

Tabela 7 - Comparação entre os classificadores baseados em regras e o algoritmo implementado

Dados	Resultados de LIM, LOH e SHIH (1999)						Resultados do Algoritmo Proposto	
	Taxa Erro			Regras			Taxa de Erro	Regras
	Mín.	Máx.	Mediana	Mín.	Máx.	Mediana		
led	0,2710	0,8160	0,2780	3	415	24	0.2850	37
led+	0,2660	0,8130	0,2900	3	13856	27	0,3165	24
bld	0,2790	0,4320	0,3220	2	31247	10	0,3775	10
bld+	0,3130	0,4410	0,3490	2	25177	6	0,3475	11
pid	0,2210	0,3100	0,2380	2	30375	7	0,2593	20
pid+	0,2210	0,3180	0,2500	2	29249	7	0,2778	19
smo	0,3040	0,4240	0,3050	1	9884	2	0,3008	42
smo+	0,3050	0,4110	0,3050	1	9718	2	0,3068	45
vot	0,0364	0,0580	0,0457	2	46695	2	0,0582	13
vot+	0,0412	0,0662	0,0469	2	31837	2	0,0611	16
Média			0,2430			9	0,2562	24

A medida de distância da Lista Longa teve um papel crucial em todos os resultados obtidos. Sua influência afetou tanto o valor da taxa de erro como número de regras geradas.

Este resultado já era esperado, uma vez que a estratégia de utilização de busca Tabu para manter a diversidade, deveria restringir a seleção freqüente de um mesmo indivíduo no processo de busca, assim como o processo de medição da distância deveria garantir que regras similares não coexistissem no mesmo conjunto de regras.

Assim, valores altos de distância sempre geraram um menor número de regras. A questão foi encontrar uma regra geral para determinar este parâmetro. Domínios com características de valores contínuos foram os que apresentaram maior dificuldade para a definição deste parâmetro. Este fato ocorreu principalmente pela medida da distância ser calculada pela diferença fenotípica dos genes.

Já nos domínios estritamente discretos, a determinação deste valor foi basicamente influenciada pelo total de características adicionadas. Este fato foi confirmado em testes realizados em conjuntos de dados com ruídos. Nessas bases, a distância teve que ser aumentada proporcionalmente ao número de novas características introduzidas, em relação ao conjunto original. Além disso, observou-se que o algoritmo não teve sua taxa de erro significativamente alterada comparando bases com e sem ruídos.

Quando analisada a questão do tempo de execução do algoritmo, verificou-se que o mesmo foi afetado principalmente pelo domínio (número de exemplos de treinamento e número de atributos contínuos e discretos), número de gerações e tamanho da população.

No sentido de tentar reduzir o custo computacional do algoritmo, testes foram realizados através da redução do tamanho da população. Para o algoritmo sem a utilização de listas de Tabu, bases como pid e led, tiveram resultados da taxa de erro comprometidos com a redução do tamanho da população. Ao se utilizar a estratégia de diversidade (busca Tabu), o algoritmo conseguiu manter os níveis de taxa de erro de classificação, sem uma alteração significativa. Um exemplo deste resultado é apresentado no Gráfico 2. A variação do tamanho do conjunto de dados Led (500, 1000, 1500 e 2000), gerou apenas uma variação de 4,6% na taxa de erro de classificação.

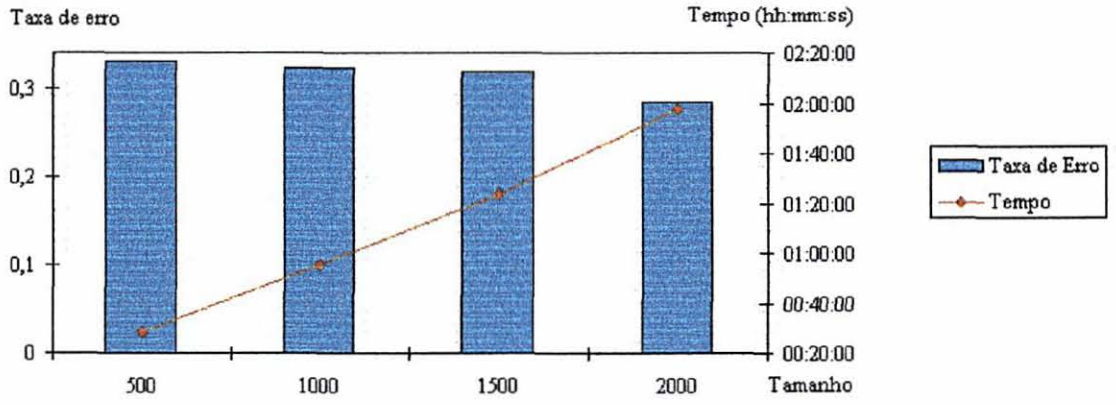

Gráfico 2 - Comportamento do tempo de processamento e taxa de erro *versus* tamanho da população

Durante os experimentos realizados, a Lista Curta não apresentou influência nos resultados, não sendo dispensados grandes esforços para a obtenção de uma regra geral para determinação do tamanho da Lista Curta.

Comparou-se também o tempo de processamento do algoritmo com e sem a utilização das listas Tabu. Estas comparações mostraram não ocorrer nenhum acréscimo significativo no tempo de processamento do algoritmo.

7. Conclusões

Neste trabalho foi apresentado um sistema para Mineração de Dados utilizando um Algoritmo Genético associado a listas Tabu. Mais especificamente, o método de Listas Tabu é utilizado como uma forma a manter a diversidade da população dos Algoritmos Genéticos, conseguindo-se, desta forma, obter uma diversidade de regras potenciais que originarão o classificador.

A ferramenta foi comparada com 34 algoritmos diferentes, analisados em LIM, LOH e SHIH (1999) e HASSE (2000). Através de cinco bases de dados, avaliou-se a precisão do classificador, o tempo de execução do algoritmo, o número de regras obtidas, a utilização de populações menores e, o comportamento do algoritmo frente à presença de ruídos.

A partir destes experimentos observou-se que:

- Não houve diferença significativa nas taxas de erro de classificação do algoritmo proposto, utilizando as listas Tabu.
- O classificador formado por um conjunto de regras no formato SE-ENTÃO, permitiu maior compreensão e clareza dos resultado obtidos.
- Através da utilização do Algoritmo Genético associado à busca Tabu, conseguiu-se manter de forma eficiente a diversidade da população, permitindo trabalhar com populações menores. Esta característica reduziu, por conseqüência, o tempo de processamento do algoritmo.
- A função de medida de distância utilizada no sistema não apresentou bons resultados quando aplicados à domínios com variáveis contínuas. Medidas de distâncias entre indivíduos são um tópico em aberto em algoritmos genéticos.
- Quando utilizada a ferramenta implementada no tratamento de dados com ruídos, o algoritmo se comportou de forma robusta.

A utilização de Algoritmos Genéticos e busca Tabu em Mineração de Dados é viável e conduz a resultados interessantes. Porém, para se obter uma conclusão favorável à esta técnica de manutenção de diversidade populacional em detrimento de outras como

criação de nichos (HASSE, 2000), torna-se necessária a realização de novos experimentos, considerando medidas de distâncias equivalentes.

As limitações e deficiências apresentadas, além das melhorias, podem ser exploradas em trabalhos futuros.

7.1. Trabalhos futuros

Há diversas etapas do algoritmo proposto que podem ser melhoradas em diversos aspectos. Os principais melhoramentos propostos para trabalhos futuros são:

- Testar e comparar outras heurísticas para a ordenação e escolha das regras.
- Unir técnicas de junção de regras de outros algoritmos, como a heurística do Algoritmo CN2 (CLARK e NIBBLET, 1989), que buscam uma regra por vez e excluem os exemplos cobertos pelas regras já usadas. Isto implica em recalcular a função de aptidão dos indivíduos restantes a cada regra adicionada, o que torna a montagem do classificador muito mais lenta.
- Utilizar outras funções de medida de distância de Lista Tabu, que consigam tratar de forma mais adequada variáveis contínuas.
- Estudar a utilização da inserção de um critério de aspiração no algoritmo, possibilitando que em certas situações uma regra possa ser reutilizada na lista Tabu, quando a mesma está totalmente preenchida.
- Analisar o comportamento do algoritmo proposto frente a outras funções *fitness*. Em HASSE (2000), foi utilizada uma função vinculada a um fator de abrangência dos indivíduos, demonstrando bons resultado.
- Adaptar a ferramenta para execução em vários processadores, discutido em FREITAS e LAVINGTON, 1998.
- Projetar novos experimentos, analisando o comportamento do algoritmo proposto frente a um maior número de conjuntos de dados.

- Identificar com maior clareza, características das bases de dados que influenciam no processo de mineração de dados através do Algoritmo Genético implementado.

- Utilizar as regras encontradas para outras aplicações de mineração de dados, como a descoberta de regras "interessantes" (NODA; FREITAS; LOPES, 1999).

Referências

ADRIAANS, P.; ZANTINGE, D. **Data mining**. England: Addison Wesley Longman, 1996.

AGRESTI, A. **Categorical data analysis.** New York: J. Wiley, 1990.

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

BAKER, J. E. Reducing bias and inefficiency in the selection algorithm. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 2., 1987. **Proceedings...** . Erlbaum: 1987.

BECKER, R. A.; CHAMBERS, J. M.; WILKS, A. R. **The new S language**. Wadsworth, 1988.

BETHKE, A. D. **Genetic algorithms as function optimizers.** 1980. Thesis, University of Michigan, Ann Arbor (Dissertation Abstracts International, 41(9), 35003B, No. 8106101. University Microfilms), 1980.

BLAKE, C. L.; MERZ, C. J. UCI **Repository of machine learning data bses**. 1998. Disponível em: <http://www.ics.uci.edu/~mlearn/MLRepository.html> Acesso em: 09 maio 2000.

BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. **Classification and regression trees.** New York: Chapman and Hall, 1984.

BRODLEY, C. E.; UTGOFF, P. E. Multivariate decision trees. **Machine Learning**, Boston, v. 19, p. 45-77, 1995.

BUNTINE, W. Learning classification trees. **Statistics and Computing**, London, v. 2, p. 63-73, 1992.

CARUANA, R. A.; SCHAFFER, J. D. Representation and hidden bias: Gry vs. binary coding for genetic algorithms. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 5., 1988. **Proceedings...** . Morgan Kaufmann, 1888.

CHEN, M. S., HAN; J., YU, P. S. Data mining: an overview from database perspective. **IEEE Transaction on Knowledge and Data Engineering**, New York, v. 8, n. 6, p. 866-883, 1996.

CLARK, L. A.; PREGIBON, D. Tree-based models. In: CHAMBERS, J. M.; HASTIE, T. J. (Eds.), **Statistical models in S.** New York: Chapman & Hall, 1993. p. 377-419.

CLARK, P.; BOSWELL, R. Rule induction with CN2: some recent improvements. In: EUROPEAN CONFERENCE ON MACHINE LEARNING, 5., 1991. **Proceedings...** . Berlin, 1991, p. 151-163.

CLARK, P.; NIBLETT, T. The CN2 induction algorithm. **Machine Learning**, Netherlands, n. 3/4, p. 261-283, 1989.

DAVIS, L. **Genetic algorithms and simulated annealing**. Pitman,1987.

DAVIS, L. **HandBook of genetic algorithms**. Nostrand Reinhold, 1991.

De JONG, K. A. **An analysis of the behavior of a class of genetic adaptive systems.** Ann Arbor, 1975. Thesis. University of Michigan.

DEB, K.; GOLDBERG, D. An investigation of niches and species formation in genetic function optimization. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 3., 1989. **Proceedings...** . San Francisco, CA: Morgan Kaufman, 1989. p. 42-50.

DEVLIN, B. **Data warehouse:** from architecture to implementation. Addinson Wesley Longman. 1997. 432p.

FAMILI, A.; SHEN, W. M.; WEBER, R.; SIMOUDIS, E. Data preprocessing and intelligent data analysis, Intell. **Data Anal**. v.1, n. 1, 1997.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P., UTHURUSAMY, R. **Advances in knowledge discovery and data mining.** Menlo Park: AAAI, 1996.

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G.; MATHEUS, C. J. Knowledge discovery in databases: an overview. In: THE AAAI WORKSHOP ON KNOWLEDGE DISCOVERY IN DATABASES. 1991. p.1- 27.

FREITAS, A; LAVINGTON, S. H. **Mining very large databases with parallel processing.** Boston: Kluwer Academic, 1998. 208p.

GIORDANA, A.; NERI, F. Search-intensive concept induction in evolutionary computation. v.3, n.4, 1995. p. 375-416.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers and Operations Research**, v 13, p. 533-549, 1986.

GOLDBERG, D. E. **Genetic algorithms in search, optimization and machine learning**. Alabama: Addison-Wesley, 1989. 413p.

GOLDBERG, D. E.; DEB, K. A comparative analysis of selection schemes used in genetic algorithms. In: RAWLINS, G. (Ed.), **Foundations of Genetic Algorithms.** San Francisco, CA: Morgan Kaufmann. 1991.

GOLDBERG, D. E.; RICHARDSON, J. Genetic algorithms with sharing for multimodal function optimization. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 2., 1987. **Proceedings...** . 1987.

GREFENSTETTE, J. J. Genetic algorithms and their applications. In: KEN, A.; WILLIANS, J. G. (Eds.), **Encyclopaedia of Computer Science and Technology.** Marcel Dekker, 1990. p. 139-152.

GREFENSTETTE, J. J. Optimization of control parameters for genetic algorithms. In: **IEEE Transaction on System, Man and Cybernetics**, New York, v. 16, p. 122-128, 1986.

HASSE, M. **Mineração de dados usando algoritmos genéticos**. Curitiba, 2000. 76p. Dissertação (Mestrado em Informática), Universidade Federal do Paraná.

HASSE, M.; POZO, A. R. Using phenotypic sharing in a classifier tool. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE, **Proceedings...** . 2000, p. 392.

HASTIE, T.; BUJA, A.; TIBSHIRANI, R. Flexible discriminant analysis by optimal scoring. **Journal of the American Statistical Association**., Boston, v. 89, p. 1255-1270, 1994.

HASTIE, T.; BUJA, A.; TIBSHIRANI, R. Penalized discriminant analysis. **Annals of Statistics**, Hayward, v. 23, p. 73-102, 1995.

HASTIE, T.; TIBSHIRANI, R. Discriminant analysis by gaussian mixtures. **Journal of the Royal Statistical Society**, London, Series B, v. 58, p. 155-176, 1996.

HOLLAND, J. H. **Adaptation in natural and artificial systems.** Michigan: University of Michigan, 1975.

HOLLAND, J. H. Escaping Brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based systems. In: MICHALSKI, R.; CARBONELL J.; MITCHELL, T. (Eds.), **Machine learning**: an AI approach. Los Altos: Morgan Kaufmann,1986. v. 2, p. 593-623.

HOLSHEIMER, M.; KERSTEN, M.; SIEBES, A. Data surveyor: searching the nuggets in parallel. In: FAYYAD, U. M. et al. (Eds.), **Advances in knowledge discovery and data mining.** Menlo Park: AAAI, 1996. p. 447-467.

HOLTE, R. C. Very simple classification rules perform well on most commonly used datasets. **Machine Learning**, Boston, v. 11, p. 63-90, 1993.

HORN, J.; DEB, K.; GOLDBERG, D. E. Implicit niching in a learning classifier system: Nature's way. **Evolutionary Computation**, v. 2, 1994. p. 37-66. (Illigal Report No. 94001 in Illinois Genetic Algorithm Laboratory, University of Illinois)

HORNE, S.; MACBETH, C. A comparation of global optimisation methods for near-offset VSP inversion. **Computers and Geosciences**, v. 24, n. 8, p. 563-572, 1998.

KELLY Jr., J. D.; DAVIS, L. A hybrid genetic algorithm for classification. **Proc. 12th IJCAI-95**, p. 645-650, 1991.

KITANO, H. Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms. **Physica D**, Amsterdam, v. 75, p. 225- 238. 1994.

KNIGHT, L.; SEM, S. PLEASE: a prototype learning system using genetic algoritms. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 6., **Proceedings...** . 1995. p. 429-435..

KOHONEN, T. **Self-organizing maps.** Heidelberg: Springer-Verlag, Heidelberg, 1995.

KOOPERBERG, C.; BOSE, S.; STONE, C. J. Polychotomous regression. **Journal of the American Statistical Association**, Boston, v. 92, p. 117-127, 1997.

KRISHNAMACHARI, B. **Global optimization in the design of mobile communication systems**. Ithaca, 1999. 175p. Master of Science in Electrical Engineering, Cornell University.

KURAHASHI, S.; TERANO, T. A Genetic algorithm with tabu search for multimodal and multiobjective function optimization. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE, 2000. **Proceedings...** . 2000. p. 291-298.

LANGLEY, P. **Elements of machine learning**. San Francisco, CA: Morgan Kaufmann, 1996. 419p.

LIM, T. S., LOH, W. Y; SHIH, Y. S. A comparison of prediction accuracy, complexity and training time of 33 old and new classification algorithms. **Machine Learning Journal**, Boston, 1999.

LOH, W. Y.; SHIH, Y. S. Split selection methods for classification trees. **Statistica Sinica**, v. 7, p. 815-840, 1997.

LOH, W. Y.; VANICHSETAKUL, N. Tree-structured classification via generalized discriminant analysis (with discussion). **Journal of the American Statistical Association**, Boston, v. 83, p. 715-728, 1988.

LOPES, F. M.; POZO, A. T. R.; VERGÍLIO, S. R.; DUARTE, D. Sistemas baseados em indução: uma comparação empírica. In: CONGRESSO LATINO-IBEROAMERICANO DE INVESTIGACIÓN DE OPERACIONES Y SISTEMAS, 10., 2000. **Resumos...** . México, 2000, 1 CD-Room.

MAHFOUD, S. W. Crowding and preselection revisited. In: MANNER, R.; MANDERICK, B. (Eds.), **Parallel Problem Solving from Nature**, 2. North-Holland, 1992. p. 27-36.

MANNILA, H. Data mining: machine learning, statistics, and databases. In: INTERNATIONAL CONFERENCE ON SCIENTIFIC AND STATISTICAL DATABASE MANANGEMENT, 8., 1996. **Proceedings...** . Stockholm, 1996. p. 1-8.

MARTÍNEZ-ENRÍQUEZ, A. M., ESCHALADA-IMAZ, G. The revision of inductive learning theory within incomplete and imprecise observations. **Expert Systems with Applications**, Tonbridge Kent, v. 15, n. 3-4, p. 357-366, 1998.

McCALLUM, R. A.; SPACKMAN, K. A. Using genetic algorithm to learn disjunctive rules from examples. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 1990. **Proceedings...** . Austin, 1990, p. 149-152.

MEYER, T. P.; PACKARD, N. H. Local forecasting of high-dimensional chaotic dynamics. In: CASDAGLI, N.; EUBANK, S. (Eds.), **Nonlinear Modeling and Forecasting.** Addison-Wesley. 1992.

MITCHELL, M. **An introduction to genetic algorithms**. Cambridge: Mit Press. 1997. 207 p.

MONTANA, D. J.; DAVIS, L. D. Training feedforward networks using genetic algorithms. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1989. **Proceedings...** . Morgan Kaukmann. 1989.

MULLER, W.; WYSOTZKI, F. Automatic construction of decision trees for classification. **Annals of Operations Research**, Amsterdam, v. 52, p. 231-247, 1994.

MULLER, W.; WYSOTZKI, F. The decision tree algorithm CAL5 based on a statistical approach to its splitting algorithm. In: NAKHAEZADEH, G.; TAYLOR, C. C. (Eds.), **Machine learning and statistics**: the interface, New York: J. Wiley, 1997, p. 45-65.

MURTHY, S. K.; KASIF, S.; SALZBERG, S. A system for induction of oblique decision trees. **Journal of Artificial Intelligence Research**, Charlottesville, v. 2, p. 1-33, 1994.

NIBLETT, T. Constructing decision trees in noisy domains. In: EUROPEAN WORKING SESSION ON LEARNING, 2., 1987. **Proceedings...** . Wilmslow, 1987. p. 67-78.

NICOLETTI, M. C.; SANTOS, F. O. Aprendizado simbólico de máquina na aquisição automática de conhecimento em domínios médicos. In: CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE, 5., **Resumos...** . Campos do Jordão, 1996. p 709-710.

NODA, E.; FREITAS, A. A.; LOPES, H. S. Discovering interesting prediction rules with a genetic algorithm. In: CONGRESS ON EVOLUTIONARY COMPUTATION (CEC-99), 1999. **Proceedings...** . Washington D. C.,1999, p. 1322-1329.

QUINLAN, J. R. **C4.5:** Programs for machine learning. San Mateo: Morgan Kaufmann, 1993. 302p.

QUINLAN, J. R. Improved use of continuous attributes in C4.5. **Journal of Artificial Intelligence Research**, Charlottesville, v. 4, p. 77-90, 1996.

SAS Institute Inc. **SAS/STAT user's guide**: version 6. v. 1 /2., 1990.

SCHULZE-KREMER, S. Genetic algorithms for protein tertiary structure prediction. In: MÄNNER, R.; MANDERICK, B. (Eds.), **Parallel Problem Solving from Nature, 2**. Amsterdam: North Holland, 1992.

SILBERSCHATZ, A.; STONEBRAKER, M.; ULLMAN, J. D. Database research: achievements and opportunities into the 21st century. In: REPORT OF AN NSF WORKSHOP ON THE FUTURE OF DATABASES SYSTEMS RESEARCH, 1995.

SMITH, S. Flexible learning of problem solving heuristics through adaptative search. In: INTERNATIONAL CONFERENCE ON ATRIFICIAL INTELLIGENCE, 8., 1983. **Proceedings...** . Karlsruhe, 1983, p. 422-425.

STATLIB **System for distributing statistical software, datasets, and information by electronic mail, FTP and WWW.StatLib.** 1994. Disponível em: <http://lib.stat.cmu.edu/datasets/csb/> Acesso em: 09 maio 2000.

STEINER, M. T. A. **Uma metodologia para o reconhecimento de padrões multivariados com resposta dicotômica**. Florianópolis, 1995. 158p. Tese (Doutorado em Engenhaia de Produção), Universidade Federal de Santa Catarina.

WATSON, J. P. **A performance assessment of modern niching methods for parameter optimization problems**. Fort Collins: Colorado State University, 1998.

WILSON, S. Classifier systems and the animat problem. **Machine Learning**, 2. Kluwer Acad., 1987, p. 199-228.

YONG, C. **Job shop scheduling with genetic algorithms and tabu search**. Massachusetts, 1994. Master of Science in Computer Science, University of Massachusetts.

ANEXOS

ANEXO 1 – Conjunto de Regras da População.

Fitness	Regras
0.920	1: IF (v3 = 2) and (v6 = 1) and (v8 = 2) and (v10 = 1) and (v16 = 2) then class = 2
0.947	2: IF (v1 = 2) and (v4 = 1) and (v8 = 2) and (v10 = 2) and (v11 = 2) and (v13 = 1) and (v15 = 2) then class = 2
0.978	3: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) and (v14 = 1) then class = 2
0.966	4: IF (v1 = 2) and (v4 = 1) and (v7 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.714	5: IF (v2 = 2) and (v3 = 2) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.937	6: IF (v2 = 1) and (v4 = 1) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v16 = 2) then class = 2
0.986	7: IF (v3 = 2) and (v7 = 2) and (v9 = 2) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.985	8: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v13 = 1) then class = 2
0.935	9: IF (v1 = 2) and (v3 = 2) and (v5 = 1) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.938	10: IF (v4 = 1) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v14 = 1) then class = 2
0.666	11: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v9 = 2) and (v10 = 1) and (v11 = 3) and (v12 = 1) and (v16 = 1) then class = 2
0.947	12: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v9 = 2) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.937	13: IF (v3 = 2) and (v4 = 1) and (v6 = 1) and (v8 = 2) and (v10 = 2) and (v14 = 1) then class = 2
0.947	14: IF (v1 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v10 = 2) and (v11 = 2) and (v15 = 2) and (v16 = 2) then class = 2
0.972	15: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.75	16: IF (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 3) and (v11 = 2) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.988	17: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v13 = 1) then class = 2
0.666	18: IF (v1 = 2) and (v3 = 2) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v15 = 3) then class = 2
0.941	19: IF (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v15 = 2) then class = 2
0.966	20: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v16 = 1) then class = 2
0.972	21: IF (v1 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.5	22: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v7 = 1) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.922	23: IF (v4 = 1) and (v6 = 1) and (v7 = 2) and (v13 = 1) then class = 2
0.969	24: IF (v3 = 2) and (v5 = 1) and (v6 = 1) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v16 = 2) then class = 2
0.849	25: IF (v5 = 1) and (v8 = 2) and (v9 = 2) and (v16 = 2) then class = 2
0.976	26: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) and (v10 = 1) and (v13 = 1) then class = 2
0.941	27: IF (v1 = 2) and (v3 = 2) and (v5 = 1) and (v7 = 2) and (v9 = 2) and (v10 = 2) and (v15 = 2) and (v16 = 1) then class = 2
0.666	28: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 3) and (v8 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.666	29: IF (v4 = 1) and (v5 = 1) and (v9 = 3) and (v13 = 1) and (v15 = 2) then class = 2
0.977	30: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v9 = 2) and (v14 = 1) and (v16 = 1) then class = 2
0.986	31: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v9 = 2) and (v11 = 2) and (v15 = 2) then class = 2
0.906	32: IF (v1 = 2) and (v4 = 1) and (v8 = 2) and (v10 = 2) and (v12 = 1) then class = 2
0.976	33: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v9 = 2) and (v13 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.949	34: IF (v3 = 2) and (v4 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v11 = 2) and (v15 = 2) and (v16 = 2) then class = 2
0.833	35: IF (v5 = 1) and (v6 = 1) and (v7 = 1) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v14 = 1) then class = 2
0.928	36: IF (v2 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.857	37: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 2) and (v13 = 1) and (v15 = 2) then class = 2

Fitness	Regras
0.966	38: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v12 = 1) and (v13 = 1) and (v16 = 1) then class = 2
0.980	39: IF (v4 = 1) and (v6 = 1) and (v11 = 2) and (v13 = 1) and (v16 = 1) then class = 2
0.936	40: IF (v1 = 2) and (v3 = 2) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v15 = 2) then class = 2
0.967	41: IF (v1 = 2) and (v3 = 2) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.5	42: IF (v1 = 2) and (v3 = 1) and (v5 = 1) and (v8 = 2) and (v10 = 2) and (v11 = 2) and (v13 = 1) and (v16 = 2) then class = 2
0.980	43: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v9 = 2) and (v10 = 1) and (v12 = 1) then class = 2
0.857	44: IF (v1 = 2) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v13 = 1) then class = 2
0.974	45: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v12 = 1) and (v13 = 1) then class = 2
0.949	46: IF (v1 = 2) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v14 = 1) then class = 2
0.941	47: IF (v1 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v16 = 2) then class = 2
0.976	48: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v8 = 2) and (v10 = 1) and (v11 = 2) then class = 2
0.973	49: IF (v1 = 2) and (v4 = 1) and (v6 = 1) and (v9 = 2) and (v11 = 2) and (v13 = 1) and (v14 = 1) then class = 2
0.978	50: IF (v4 = 1) and (v9 = 2) and (v11 = 2) and (v15 = 2) then class = 2
0.941	51: IF (v1 = 2) and (v2 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v10 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.937	52: IF (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 2) and (v11 = 2) and (v14 = 1) and (v16 = 1) then class = 2
0.956	53: IF (v2 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.947	54: IF (v1 = 2) and (v2 = 1) and (v4 = 1) and (v7 = 2) and (v9 = 2) and (v13 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.977	55: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.764	56: IF (v4 = 1) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v16 = 3) then class = 2
0.884	57: IF (v1 = 2) and (v9 = 2) and (v10 = 1) and (v12 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.944	58: IF (v1 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v13 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.989	59: IF (v3 = 2) and (v4 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.666	60: IF (v1 = 2) and (v4 = 1) and (v7 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.958	61: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v16 = 1) then class = 2
0.949	62: IF (v6 = 1) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.911	63: IF (v2 = 2) and (v3 = 2) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v13 = 1) and (v15 = 2) then class = 2
0.982	64: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) then class = 2
0.936	65: IF (v3 = 2) and (v11 = 2) and (v13 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.985	66: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v11 = 2) and (v13 = 1) and (v14 = 1) then class = 2
0.941	67: IF (v1 = 2) and (v4 = 1) and (v7 = 2) and (v12 = 1) and (v15 = 2) then class = 2
0.972	68: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) then class = 2
0.888	69: IF (v1 = 2) and (v3 = 2) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v13 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.980	70: IF (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.970	71: IF (v1 = 2) and (v4 = 1) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.968	72: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v10 = 1) and (v14 = 1) and (v16 = 1) then class = 2
0.935	73: IF (v4 = 1) and (v6 = 1) and (v10 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.988	74: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v9 = 2) and (v11 = 2) then class = 2
0.972	75: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) then class = 2
0.976	76: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.969	77: IF (v1 = 2) and (v4 = 1) and (v8 = 2) and (v11 = 2) and (v15 = 2) and (v16 = 2) then class = 2
0.333	78: IF (v1 = 2) and (v3 = 2) and (v5 = 1) and (v7 = 1) and (v8 = 2) and (v10 = 2) and (v11 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.903	79: IF (v1 = 2) and (v2 = 2) and (v3 = 2) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v13 = 1) and (v15 = 2) then class = 2
0.971	80: IF (v2 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v11 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2

Fitness	Regras
0.941	81: IF (v2 = 1) and (v3 = 2) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.933	82: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.979	83: IF (v4 = 1) and (v7 = 2) and (v9 = 2) and (v10 = 1) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.916	84: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v11 = 1) and (v12 = 1) and (v13 = 1) then class = 2
0.75	85: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 3) and (v12 = 1) and (v13 = 1) and (v15 = 2) then class = 2
0.984	86: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.666	87: IF (v1 = 2) and (v3 = 2) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 1) then class = 2
0.968	88: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 1) and (v14 = 1) then class = 2
0.875	89: IF (v1 = 2) and (v4 = 1) and (v7 = 1) and (v9 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.933	90: IF (v1 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v11 = 2) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.928	91: IF (v2 = 1) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) and (v10 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.75	92: IF (v1 = 2) and (v4 = 1) and (v8 = 2) and (v9 = 2) and (v11 = 3) and (v12 = 1) and (v15 = 2) then class = 2
0.971	93: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v10 = 2) and (v11 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.800	94: IF (v1 = 2) and (v4 = 1) and (v6 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.968	95: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.968	96: IF (v1 = 2) and (v3 = 2) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 1) and (v15 = 2) then class = 2
0.857	97: IF (v1 = 2) and (v4 = 1) and (v6 = 1) and (v8 = 3) and (v14 = 1) then class = 2
0.965	98: IF (v3 = 2) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v11 = 2) then class = 2
0.666	99: IF (v4 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v13 = 1) and (v15 = 3) and (v16 = 3) then class = 2
0.979	100: IF (v1 = 2) and (v4 = 1) and (v10 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.981	101: IF (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) then class = 2
0.947	102: IF (v1 = 2) and (v2 = 2) and (v3 = 2) and (v4 = 1) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v15 = 2) then class = 2
0.985	103: IF (v4 = 1) and (v10 = 1) and (v11 = 2) and (v14 = 1) then class = 2
0.977	104: IF (v4 = 1) and (v8 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v16 = 1) then class = 2
0.976	105: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v16 = 1) then class = 2
0.5	106: IF (v3 = 2) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 2) and (v14 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.974	107: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v8 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.988	108: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v11 = 2) then class = 2
0.971	109: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v9 = 2) and (v15 = 2) then class = 2
0.974	110: IF (v4 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v14 = 1) then class = 2
0.971	111: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v15 = 2) then class = 2
0.980	112: IF (v4 = 1) and (v5 = 1) and (v9 = 2) and (v11 = 2) and (v14 = 1) then class = 2
0.962	113: IF (v1 = 2) and (v4 = 1) and (v13 = 1) and (v16 = 1) then class = 2
0.981	114: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v10 = 1) and (v15 = 2) then class = 2
0.666	115: IF (v2 = 3) and (v3 = 2) and (v4 = 1) and (v7 = 1) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.974	116: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v16 = 2) then class = 2
0.972	117: IF (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v16 = 1) then class = 2
0.970	118: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v11 = 2) and (v13 = 1) then class = 2
0.666	119: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v9 = 1) then class = 2
0.969	120: IF (v4 = 1) and (v13 = 1) and (v16 = 1) then class = 2
0.800	121: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 1) and (v8 = 2) and (v9 = 2) and (v14 = 1) then class = 2

Fitness	Regras
0.958	122: IF (v1 = 2) and (v3 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v13 = 1) and v14 = 1) then class = 2
0.987	123: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.971	124: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v9 = 2) and (v12 = 1) and (v13 = 1) and (v15 = 2) then class = 2
0.977	125: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v10 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.977	126: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v9 = 2) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.833	127: IF (v1 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v12 = 3) and (v13 = 1) and (v14 = 1) then class = 2
0.987	128: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v9 = 2) and (v11 = 2) and (v15 = 2) then class = 2
0.987	129: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v9 = 2) and (v11 = 2) and (v12 = 1) then class = 2
0.666	130: IF (v3 = 2) and (v4 = 1) and (v8 = 3) and (v11 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.941	131: IF (v1 = 2) and (v2 = 2) and (v3 = 2) and (v4 = 1) and (v7 = 2) and (v10 = 2) and (v12 = 1) and (v13 = 1) and (v15 = 2) then class = 2
0.972	132: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v16 = 2) then class = 2
0.5	133: IF (v5 = 1) and (v7 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v13 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.977	134: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v12 = 1) then class = 2
0.800	135: IF (v3 = 2) and (v4 = 1) and (v7 = 1) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.937	136: IF (v4 = 1) and (v5 = 1) and (v8 = 2) and (v15 = 2) then class = 2
0.800	137: IF (v4 = 1) and (v8 = 3) and (v11 = 2) and (v13 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.937	138: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v9 = 2) and (v10 = 2) and (v15 = 2) then class = 2
0.977	139: IF (v4 = 1) and (v5 = 1) and (v9 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.928	140: IF (v1 = 2) and (v2 = 1) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 1) and (v15 = 2) then class = 2
0.984	141: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v13 = 1) then class = 2
0.977	142: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.905	143: IF (v2 = 1) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v12 = 1) and (v15 = 2) then class = 2
0.979	144: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v10 = 1) and (v13 = 1) and (v14 = 1) then class = 2
0.666	145: IF (v4 = 1) and (v7 = 2) and (v9 = 2) and (v10 = 2) and (v12 = 1) and (v13 = 2) and (v16 = 1) then class = 2
0.5	146: IF (v2 = 1) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v14 = 2) and (v15 = 2) then class = 2
0.949	147: IF (v2 = 1) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v13 = 1) then class = 2
0.916	148: IF (v1 = 2) and (v2 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.5	149: IF (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 1) and (v12 = 1) and (v13 = 2) and (v15 = 2) then class = 2
0.964	150: IF (v1 = 2) and (v4 = 1) and (v8 = 2) and (v13 = 1) and (v15 = 2) then class = 2
0.951	151: IF (v4 = 1) and (v6 = 1) and (v7 = 2) and (v9 = 2) and (v10 = 2) and (v12 = 1) and (v13 = 1) then class = 2
0.967	152: IF (v1 = 2) and (v4 = 1) and (v6 = 1) and (v11 = 2) and (v14 = 1) then class = 2
0.800	153: IF (v2 = 3) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v12 = 1) and (v13 = 1) then class = 2
0.954	154: IF (v1 = 2) and (v2 = 1) and (v4 = 1) and (v9 = 2) and (v16 = 1) then class = 2
0.973	155: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) then class = 2
0.972	156: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v15 = 2) then class = 2
0.975	157: IF (v1 = 2) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v14 = 1) then class = 2
0.875	158: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v9 = 2) and (v10 = 1) and (v13 = 2) and (v14 = 1) then class = 2
0.957	159: IF (v1 = 2) and (v4 = 1) and (v5 = 1) and (v12 = 1) and (v13 = 1) then class = 2
0.937	160: IF (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v15 = 2) then class = 2
0.955	161: IF (v4 = 1) and (v5 = 1) and (v9 = 2) and (v13 = 1) then class = 2
0.987	162: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v13 = 1) and (v14 = 1) then class = 2

Fitness	Regras
0.939	163: IF (v1 = 2) and (v3 = 2) and (v5 = 1) and (v6 = 1) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.5	164: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 3) and (v12 = 1) and (v13 = 1) then class = 2
0.973	165: IF (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v11 = 2) and (v12 = 1) and (v14 = 1) then class = 2
0.857	166: IF (v1 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 1) and (v10 = 1) and (v11 = 2) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.979	167: IF (v4 = 1) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v14 = 1) then class = 2
0.5	168: IF (v1 = 2) and (v2 = 2) and (v4 = 1) and (v7 = 3) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v14 = 1) then class = 2
0.833	169: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 3) and (v14 = 1) and (v15 = 2) then class = 2
0.977	170: IF (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v16 = 1) then class = 2
0.857	171: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v10 = 1) and (v11 = 1) and (v12 = 1) and (v13 = 1) then class = 2
0.952	172: IF (v2 = 1) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v9 = 2) then class = 2
0.978	173: IF (v3 = 2) and (v4 = 1) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v13 = 1) then class = 2
0.972	174: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v9 = 2) and (v13 = 1) and (v14 = 1) and (v16 = 2) then class = 2
0.966	175: IF (v1 = 2) and (v2 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 1) and (v15 = 2) then class = 2
0.933	176: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v11 = 2) and (v12 = 1) and (v13 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.944	177: IF (v1 = 2) and (v3 = 2) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v15 = 2) and (v16 = 1) then class = 2
0.985	178: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v12 = 1) and (v13 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.962	179: IF (v3 = 2) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v11 = 2) and (v12 = 1) then class = 2
0.924	180: IF (v3 = 2) and (v6 = 1) and (v7 = 2) and (v10 = 1) and (v13 = 1) then class = 2
0.857	181: IF (v1 = 2) and (v4 = 1) and (v5 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v16 = 3) then class = 2
0.962	182: IF (v4 = 1) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v12 = 1) then class = 2
0.5	183: IF (v1 = 2) and (v6 = 1) and (v7 = 2) and (v8 = 3) and (v11 = 2) and (v12 = 3) then class = 2
0.827	184: IF (v2 = 1) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2
0.970	185: IF (v3 = 2) and (v9 = 2) and (v11 = 2) then class = 2
0.800	186: IF (v1 = 2) and (v2 = 3) and (v3 = 2) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v12 = 1) and (v13 = 1) then class = 2
0.888	187: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 2) and (v11 = 2) and (v14 = 1) then class = 2
0.988	188: IF (v3 = 2) and (v4 = 1) and (v9 = 2) and (v11 = 2) and (v14 = 1) and (v15 = 2) then class = 2
0.947	189: IF (v3 = 2) and (v4 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v13 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 1) then class = 2
0.833	190: IF (v1 = 2) and (v2 = 1) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v9 = 2) and (v10 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.862	191: IF (v1 = 2) and (v11 = 2) and (v15 = 2) then class = 2
0.938	192: IF (v1 = 2) and (v4 = 1) and (v9 = 2) and (v10 = 1) and (v13 = 1) then class = 2
0.857	193: IF (v1 = 2) and (v8 = 2) and (v10 = 1) and (v14 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.833	194: IF (v1 = 1) and (v3 = 2) and (v4 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v12 = 1) and (v13 = 1) then class = 2
0.922	195: IF (v4 = 1) and (v5 = 1) and (v6 = 1) and (v8 = 2) then class = 2
0.913	196: IF (v3 = 2) and (v5 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v16 = 1) then class = 2
0.978	197: IF (v3 = 2) and (v4 = 1) and (v6 = 1) and (v8 = 2) and (v10 = 1) and (v11 = 2) and (v15 = 2) then class = 2
0.949	198: IF (v4 = 1) and (v5 = 1) and (v8 = 2) and (v9 = 2) and (v10 = 2) and (v12 = 1) and (v13 = 1) and (v15 = 2) and (v16 = 2) then class = 2
0.975	199: IF (v3 = 2) and (v4 = 1) and (v8 = 2) and (v10 = 2) and (v11 = 2) and (v15 = 2) then class = 2
0.976	200: IF (v1 = 2) and (v3 = 2) and (v4 = 1) and (v6 = 1) and (v7 = 2) and (v8 = 2) and (v10 = 1) and (v12 = 1) and (v14 = 1) and (v15 = 2) then class = 2