GISELE LOBO PAPPA

SELEÇÃO DE ATRIBUTOS UTILIZANDO ALGORITMOS GENÉTICOS MULTIOBJETIVOS

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

CURITIBA
2002

GISELE LOBO PAPPA

SELEÇÃO DE ATRIBUTOS UTILIZANDO ALGORITMOS GENÉTICOS MULTIOBJETIVOS

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

Área de Concentração: *Sistemas Inteligentes*

Orientador: Prof. Dr. Celso Antônio Alves Kaestner
Co-orientador: Prof. Dr. Alex Alves Freitas

CURITIBA
2002

Pappa, Gisele Lobo
Seleção de Atributos utilizando Algoritmos Genéticos Multiobjetivos.
Curitiba, 2002. 85p.

Dissertação (Mestrado) – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. Mineração de Dados 2. Seleção de Atributos 3. Algoritmos Genéticos 4. Otimização Multiobjetiva. I. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada

TERMO DE APROVAÇÃO

Agradecimentos

"Sonho que se sonha só, é só um sonho.
Sonho que se sonha junto, é realidade."

À Deus, por ter me guiado e concedido discernimento para fazer escolhas tão certas quanto as que venho fazendo.

À minha família e ao meu namorado, Márcio, por toda paciência, apoio e compreensão nesses últimos dois anos.

Aos meus queridos amigos Daniella, Cristiane, David, Aline, Evandro, Otávio, Fernanda e Díbio, pelos momentos de estudo e descontração.

À Motorola, pelo suporte financeiro concedido para que essa pesquisa fosse realizada.

Ao Prof. Maziero, pela disponibilização e ajuda com as máquinas onde os testes que constam nessa pesquisa foram realizados.

Aos professores Alex A. Freitas, Celso Kaestner e Julio César Nievola, pela paciência, orientação e longas discussões sobre os rumos deste trabalho.

E a todos aqueles que de alguma maneira contribuíram para que esse trabalho fosse realizado.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A seleção de atributos é uma das tarefas que podem ser realizadas durante a fase de pré-processamento de dados que serão posteriormente minerados. Ela é importante porque, na maioria dos casos, dados são coletados para propósitos diferentes da classificação. Por isso, bases de dados costumam conter muitos atributos irrelevantes que, se não removidos, podem tornar o processo de aprendizagem difícil.

Este trabalho propõe um algoritmo genético (AG) multiobjetivo para seleção de atributos. Seu projeto e implementação foram motivados pelo grande sucesso dos AGs em aplicações em que o espaço de busca é grande, e por apresentar a vantagem de realizar uma busca global no espaço de soluções candidatas, ao contrário de outros algoritmos baseados em busca local.

O AG proposto utiliza conceitos de otimização multiobjetiva, pois o problema da seleção de atributos requer, em nosso caso, a otimização de dois objetivos: o erro de classificação e o número de regras produzidas por um algoritmo de indução de regras.

A avaliação dos indivíduos é feita de acordo com a abordagem *wrapper*, ou seja, para cada indivíduo da população, o algoritmo de classificação a ser posteriormente utilizado é executado, a fim de tornar a seleção de atributos mais robusta. O algoritmo de *classificação* empregado pelo método é o C4.5.

Além do algoritmo genético multiobjetivo, este trabalho propõe também uma versão multiobjetiva do método de seleção seqüencial para frente, a fim de comparar versões multiobjetivas de dois métodos tradicionalmente utilizados na tarefa de seleção de atributos.

Experimentos foram realizados em 18 bases de dados de domínio público, e tanto o algoritmo genético multiobjetivo quanto o método de seleção seqüencial para frente multiobjetivo propostos mostraram-se métodos eficazes para solução do problema de seleção de atributos, e competitivos como outros algoritmos tradicionais da literatura de otimização multiobjetiva, como o SPEA (*Strenght Pareto Evolutionary Algorithm*).

Palavras-Chave: mineração de dados, seleção de atributos, algoritmos genéticos, otimização multiobjetiva

ABSTRACT

Attribute selection is one of the tasks that can be performed during the preprocessing of the data to be mined. It is an important task because, in the majority of the cases, data is collected for purposes other than classification. As a result, databases usually contain many irrelevant attributes, and if these attributes are not removed they can hinder the process of learning.

This work proposes a multiobjective Genetic Algorithm (GA) for attribute selection. Its development and implementation were motivated by the great success obtained by GAs in applications where the search space is vast and by the advantage of performing a global search in the space of candidate solutions, unlike other algorithms based on local search.

The proposed GA uses concepts of multiobjective optimization, since the attribute selection problem requires, in our case, the optimization of two objectives: the classification error and the number of rules generated by a rule induction algorithm.

The evaluation of the individuals is performed according to the the wrapper approach, i.e., the evaluation of each individual of the population involves running the classification algorithm to be used later (with the set of selected attributes), in order to make the attribute selection procedure more robust. The classification algorithm used in this work is C4.5.

In addition to the multiobjective GA, this work also proposes a multiobjective version of the forward sequential selection method, in order to compare multiobjective versions of two methods often used in the attribute selection task.

Experiments in 18 public-domain databases showed that the multiobjective genetic algorithm and the multiobjective forward feature selection algorithm proposed can solve the feature selection task better than the single objective methods.

Key-words: Data Mining, Attribute Selection, Genetic Algorithms. Multiobjective Optimization

1. INTRODUÇÃO

Em 1934, em sua obra *The Rock*, o poeta americano Thomas Stearns Eliot acrescentou a seus versos duas questões:

"*Where is the wisdom we have lost in knowledge?*

Where is the knowledge we have lost in information?"[1]

Provavelmente se essa pergunta fosse feita, nos dias de hoje, a resposta seria algo parecido com: em imensas e, muitas vezes, desconhecidas bases de dados. Isso porque dados crescem em uma velocidade muito maior que a capacidade humana de processá-los, e o maior desafio é torná-los úteis, extraindo deles informações interessantes.

A complexidade apresentada pelas imensas bases de dados que surgem, como forma de armazenar grandes quantidades de dados, levou os pesquisadores a reduzirem a dimensionalidade do problema de descoberta de conhecimento, amostrando os dados disponíveis. Essa redução pode ser feita em duas direções: pela seleção aleatória de alguns registros, que serão utilizados para exploração do espaço de dados, ou pela redução do espaço de atributos [Martín-Bautista e Vila 1999] [Blum e Langley 1997]. Obviamente, no último caso, a seleção dos atributos relevantes pode ocasionar também a redução do espaço de dados.

Essa amostragem faz sentido principalmente por bases de dados serem geradas com propósitos diferentes dos de descoberta de conhecimento e aprendizagem de máquina [Holsheimer e Siebes 1991].

Mineração de dados é hoje o termo associado à busca de conhecimento compreensível, útil e surpreendente em grandes bases de dados, e sua aplicação dispensa a presença de um número significativo de atributos ou mesmo registros presentes nas bases de dados originais, e que em certos casos, se não forem removidos, podem até "atrapalhar" o processo de aprendizagem.

Dada a importância de se reduzir o espaço de dados e atributos, pesquisas na área de seleção de atributos foram iniciadas há muito tempo nas áreas de estatística e

[1] Onde está a sabedoria que perdemos no conhecimento? Onde está o conhecimento que perdemos na informação?

reconhecimento de padrões, e só posteriormente passaram a ser tratadas na área de aprendizagem de máquina. Porém, a solução para esse problema não é trivial nem única.

A seleção de atributos tem como objetivo descobrir um subconjunto de atributos relevantes para uma tarefa alvo, considerando os atributos originais, e é importante, entre outras coisas, por tornar o processo de aprendizagem mais eficiente. Atributos redundantes prejudicam a performance do algoritmo de aprendizagem tanto na velocidade (devido à dimensionalidade dos dados) quanto na taxa de acerto (devido à presença de informações redundantes que podem confundir o algoritmo, ao invés de auxiliá-lo na busca de um modelo correto para o conhecimento) [Kira e Rendell 1992].

Quando a tarefa alvo da seleção de atributos é a classificação, a seleção de atributos normalmente busca minimizar a taxa de erro do classificador, a complexidade do conhecimento gerado por ele, e o número de atributos selecionados para compor a "nova" base.

Dessa forma, a seleção de atributos é apenas mais um dos problemas do mundo real que envolve a otimização simultânea de mais de um critério ou objetivo. Mas nem sempre essa otimização é possível. Objetivos podem ser conflitantes, e normalmente medem aspectos diferentes de um problema. No caso da seleção de atributos na tarefa de classificação, intuitivamente taxas de erros menores podem ser geradas a partir de um número de regras ou um número de condições por regra maiores.

Com o intuito de solucionar problemas com mais de um objetivo, surgiu o conceito de otimização multiobjetiva [Deb 2001]. Quando utilizado esse conceito permite que, para um determinado problema, um conjunto de soluções ótimas seja apresentado sem privilegiar um ou outro objetivo, e deixa a cargo do usuário a escolha da solução que mais se adapte às suas necessidades.

Pensando na seleção de atributos como um problema de otimização multiobjetiva, este trabalho propõe um algoritmo genético multiobjetivo para a tarefa de seleção de atributos. Algoritmos genéticos foram escolhidos por serem um método de busca robusto, capaz de explorar grandes espaços de atributos. Além disso, ao contrário da maioria dos algoritmos de busca tradicionais, eles identificam e exploram interações não lineares entre os atributos, e realizam uma busca global [Goldberg 1989]. Por último, temos que considerar que a otimização multiobjetiva exige um algoritmo capaz de gerar um conjunto de soluções ótimas a cada iteração, e essa exigência é também atendida pelos algoritmos genéticos.

O principal objetivo desse trabalho é verificar o comportamento de algoritmos genéticos multiobjetivos em tarefas relacionadas à área de seleção de atributos em mineração de dados, e foi motivado pelos ótimos resultados obtidos com a utilização de algoritmos genéticos multiobjetivos na otimização dos mais diversos problemas das mais variadas áreas, como mostrado em [Coello *et* al 2002]. Os primeiros experimentos na área de seleção de atributos com algoritmos genéticos multiobjetivos, realizados por [Emmanouilidis et al 2000] e [Kim *et* al 2000], descritos no Capítulo 6, mostram como a otimização multiobjetivo pode trazer resultados superiores aos obtidos através da otimização simples.

Além do algoritmo genético multiobjetivo, esse trabalho propõe também uma versão modificada do método de seleção seqüencial para frente, um algoritmo de busca gulosa vastamente utilizado na solução do problema de seleção de atributos. À sua versão padrão foram acrescentados conceitos de otimização multiobjetiva, da mesma forma que no algoritmo genético proposto.

Esse trabalho está organizado em sete capítulos. Os Capítulos 2 e 3 apresentam uma revisão dos principais conceitos de mineração de dados e algoritmos genéticos, respectivamente. O Capítulo 4, por sua vez, descreve a abordagem proposta para solucionar o problema da seleção de atributos utilizando múltiplos objetivos. Já o Capítulo 5 apresenta os experimentos realizados, bem como os resultados obtidos. O Capítulo 6 apresenta uma revisão bibliográfica dos trabalhos relacionados à seleção de atributos, utilizando algoritmos genéticos multiobjetivos. Por último, as conclusões do trabalho são apresentadas no Capítulo 7.

2. MINERAÇÃO DE DADOS

Há muito tempo o homem descobriu as vantagens de armazenar dados eletronicamente, e trocou o papel por meios de armazenamento mais eficientes e convenientes, como fitas e discos ópticos.

Essa mudança facilitou o acesso aos dados, e os sistemas de gerenciamento de banco de dados possibilitaram que estes fossem facilmente armazenados, manipulados ou compartilhados. Com o passar do tempo, esses bancos tornaram-se maiores, e descobriu-se que havia muito mais informação útil escondida neles que se pudesse imaginar.

A diferença entre dados e informação é ainda maior quando nos referimos a sistemas de banco de dados. Com o intuito de transformar dados em informação, e posterior conhecimento útil, surgiu o processo de descoberta de conhecimento em bases de dados, ou KDD (*Knowledge Discovery in Databases*). Associado a esse processo surgiu o termo mineração de dados (*data mining*).

Mineração de dados é definida como a busca por relacionamentos e padrões interessantes existentes em bases de dados do mundo real, mas que estão escondidos em meio a uma grande quantidade de dados armazenados. Esses relacionamentos representam conhecimento valioso sobre a base de dados e, conseqüentemente, sobre o domínio do mundo real que elas representam [Holsheimer e Siebes 1991].

A Figura 2.1 apresenta as etapas do processo de KDD, sendo uma delas denominada mineração de dados.

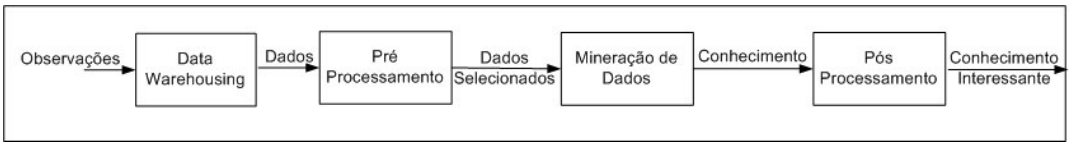


Figura 2.1: Fases do processo de KDD [Liu e Motoda 1998]

Data Warehousing, nome dado à primeira etapa do processo de KDD ilustrado pela Figura 2.1, é o processo que permite que dados de diferentes fontes e em diferentes formatos sejam coletados e depois reunidos para serem utilizados em uma mesma aplicação.

A fase de Pré-Processamento seleciona, a partir dos dados gerados pela fase de *Data Warehousing*, aqueles considerados úteis para a tarefa que se pretende realizar, dando origem a uma base de dados. A essa nova base de dados são então aplicados métodos de remoção de ruídos e de projeção e redução dos dados, remodelando-os de acordo com a aplicação em que serão utilizados.

Na etapa denominada mineração de dados, escolhe-se um algoritmo propício para a aplicação em questão, que será utilizado para procurar por padrões interessantes e úteis nos dados.

A fase de Pós-Processamento compreende a validação e interpretação dos resultados, além da organização da informação encontrada, de maneira que ela possa ser verificada e reusada.

Foi citado anteriormente que na fase da mineração de dados escolhe-se um algoritmo propício para a aplicação em questão. Nessa frase, o termo propício refere-se principalmente ao tipo de tarefa que o algoritmo será utilizado para resolver.

Dependendo do domínio da aplicação e do interesse do usuário, vários tipos de tarefas podem ser identificadas e posteriormente realizadas através do processo de KDD. Em geral, cada tarefa extrai um tipo de conhecimento diferente da base de dados, e por isso elas utilizam algoritmos diferentes.

A Seção 2.1 descreve os principais tipos de tarefas que podem ser resolvidas pelo processo de KDD.

2.1 Tarefas resolvidas pelo processo de KDD

Entre as principais tarefas resolvidas pelo processo de KDD estão a associação, a classificação e o agrupamento. As próximas seções descrevem com mais detalhes esses 3 tipos de tarefas.

2.1.1 Associação

A tarefa de associação tem como principal objetivo encontrar, a partir de um conjunto de exemplos *E*, um conjunto de regras de associação, ou seja, descobrir quais atributos aparecem freqüentemente associados nesses exemplos.

Esse tipo de tarefa é normalmente aplicado a um tipo especial de dados, denominado "cesta de mercado" (*basket data*), em que cada registro consiste de um conjunto de atributos denominados itens, geralmente binários.

Uma regra de associação é um relacionamento do tipo Se (x) então (y), onde $x \in X$ e $y \in Y$ são conjuntos de itens, e $X \cap Y = \varnothing$.

Um registro normalmente corresponde a uma transação de um cliente, em que cada item tem valor verdadeiro ou falso, de acordo com a compra ou não do item pelo cliente. Esse tipo de dado é normalmente coletado através de tecnologias, como a de código de barras [Freitas e Lavington 1998].

2.1.2 Classificação

A classificação é uma das tarefas mais comumente resolvidas com técnicas de mineração de dados. Um sistema de classificação é utilizado para prever a classe de um objeto baseado em seus atributos.

Os dados utilizados para resolução desse tipo de tarefa consistem em um conjunto de atributos denominados previsores e um atributo denominado meta, que define a classe a que esse registro pertence. O objetivo dessa tarefa é descobrir um relacionamento entre os atributos previsores e o atributo meta, usando registros cuja classe é conhecida, para que posteriormente esses atributos previsores possam ser utilizados para prever a classe de um registro cuja classe é desconhecida [Hand 1997].

Quando trabalhamos com um classificador, os exemplos disponíveis para criação de um modelo de classificação são divididos em dois conjuntos mutuamente exclusivos: um conjunto de treinamento e um conjunto de teste. O conjunto de treinamento fica disponível para o classificador, que analisa as relações entre os atributos previsores e o meta. Os relacionamentos descobertos, a partir desses exemplos, são então utilizados para prever a classe dos registros presentes no conjunto de teste. Para o classificador, o atributo meta do conjunto de teste fica indisponível. Após prever a classe dos exemplos do conjunto de teste, as classes previstas são então comparadas com as classes reais dos exemplos, definidas pelo atributo meta. Se a classe prevista for igual a real, a previsão foi correta, caso contrário, a previsão foi incorreta.

Um dos principais objetivos na tarefa de classificação é maximizar a taxa de classificações corretas nos dados de teste, que corresponde à razão entre o número de

exemplos corretamente classificados e o número total de exemplos disponíveis no conjunto de testes.

O conhecimento descoberto pelo classificador, através dos exemplos de treinamento, pode ser representado de várias formas. Neste trabalho, o interesse está voltado para o conhecimento representado através de árvores de decisão, método utilizado pelo C4.5 [Quinlan 1993].

Uma árvore de decisão é induzida a partir de um conjunto de exemplos de treinamento onde as classes são previamente conhecidas, e é formada por 3 componentes principais:

1. Nós internos, representando atributos previsores
2. Nós folhas, representando os valores que o atributo meta pode assumir
3. Ramos ou arestas, que partem de nós internos e representam um dos valores, ou um conjunto de valores (intervalos), que estes podem assumir.

Para classificar um exemplo, devemos partir do nó raiz e seguir o caminho indicado pelos ramos da árvore, de acordo com os valores de atributos previsores que o exemplo assume serem compatíveis com os indicados nos ramos ou arestas. Ao encontrar um nó folha, encontramos também a classe a que o exemplo pertence.

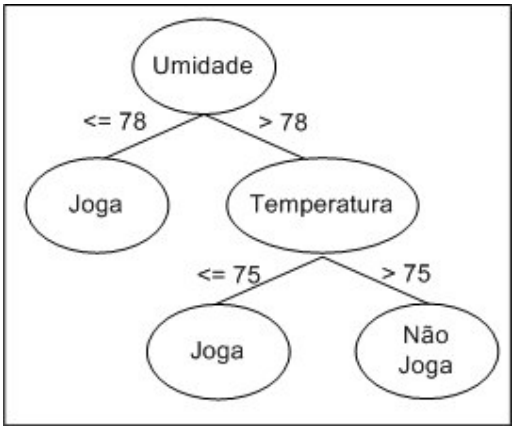
O tamanho da árvore é definido então pelo número de nós que ela possui, sejam eles nós internos ou nós folhas. A Figura 2.2 (a) mostra um exemplo de um conjunto de registros de treinamento utilizados para prever se o dia é apropriado para jogar golfe de acordo com quatro elementos: a previsão do tempo, a temperatura, a umidade e o vento. A Figura 2.2.(b) mostra a árvore criada a partir deste conjunto de treinamento [Quinlan 1993].

O tamanho da árvore da Figura 2.2 (b) é cinco, uma vez que a árvore é composta de 5 nós, e o erro nos dados de treinamento após a execução do algoritmo de aprendizagem C4.5 é 11.1%.

Dados de Treinamento

sol, 85, 85, falso, Não Joga
sol, 80, 90, verdadeiro, Não Joga
nublado, 83, 78, falso, Joga
chuva, 70, 96, falso, Joga
chuva, 68, 80, falso, Joga
sol, 75, 70, verdadeiro, Joga
nublado, 72, 90, verdadeiro, Joga
nublado, 81, 75, falso, Joga
chuva, 71, 80, verdadeiro, Não Joga

(a)



(b)

Figura 2.2: Exemplo de árvore criada pelo C4.5

2.1.3 Agrupamento

A tarefa de agrupamento divide os dados em grupos formados por elementos com características semelhantes [Fayyad *et* al 1996]. Nesse tipo de problema, o sistema deve particionar o conjunto de dados em subconjuntos.

Um algoritmo de agrupamento deve ser capaz de maximizar a semelhança entre os elementos de um mesmo grupo e minimizar as semelhanças entre exemplos pertencentes a grupos diferentes. Normalmente não existe uma resposta correta para um problema de agrupamento.

A tarefa de agrupamento possibilita um entendimento inicial dos dados, e na maioria dos casos, após o agrupamento, métodos de classificação ou sumarização são aplicados a fim de obter regras de classificação (que distinguem registros pertencentes a classes diferentes) ou regras de sumarização (que caracterizem cada grupo/classe) [Freitas 2002].

2.2 Seleção de Atributos

Intuitivamente, quanto maior o número de atributos em uma base de dados, maior o poder discriminatório do classificador e a facilidade de extrair modelos de conhecimento da base. Porém, quando a teoria vira prática, o mundo real apresenta provas de que nem sempre isso é verdade. Isso porque, em primeiro lugar, muitos métodos de indução sofrem da maldição da dimensionalidade, ou seja, o tempo computacional do algoritmo aumenta agressivamente de acordo com o número de atributos presentes.

Além disso, de certa forma os algoritmos de aprendizagem fazem uma estimativa de probabilidade de uma classe, dado um conjunto de atributos previsores. Em domínios com muitos atributos, essa distribuição é complexa. Considerando ainda que a quantidade de dados disponíveis para indução do modelo é limitada, obter boas estimativas para parâmetros probabilísticos torna-se tarefa ainda mais complicada. Por último, atributos irrelevantes ou redundantes podem confundir o algoritmo de aprendizagem, ajudando a esconder as distribuições de pequenos conjuntos de atributos realmente relevantes [Koller e Sahami 1996].

Independente da quantidade de dados ou atributos presentes em uma base de dados, a seleção de atributos é hoje uma das principais tarefas de pré-processamento utilizadas para preparar dados que serão posteriormente minerados. Ela tem como principal objetivo selecionar um subconjunto de atributos relevantes para a tarefa alvo, dentre todos os atributos disponíveis. De maneira simples, pode-se definir um atributo como relevante se ele é capaz de distinguir exemplos pertencentes a classes diferentes. Porém, na literatura existem várias definições formais para atributos relevantes, subdividindo-os em atributos de fraca e forte relevância [John *et* al 1994]. Alguns algoritmos utilizam a relevância de cada atributo para auxiliar durante a seleção, como mostrado em [Boz 2002].

A seleção de atributos tem como objetivos:

- Melhorar a performance do algoritmo de mineração de dados (velocidade de aprendizagem, taxa de classificações corretas e/ou simplicidade das regras).
- Remover ruídos e diminuir a dimensionalidade dos dados.

Além disso, experimentos comprovam que o número de exemplos utilizados para garantir uma certa taxa de classificação cresce exponencialmente com o número de atributos irrelevantes presentes [Langley e Iba 1993].

A seleção de atributos garante que os dados que chegam à fase da mineração sejam de boa qualidade [Liu e Motoda 1998]. Além disso, a seleção de atributos também é útil quando temos muitos atributos ou poucos registros.

Os algoritmos utilizados para seleção de atributos podem ser normalmente divididos em 2 etapas:

1. Busca dos subconjuntos de atributos
2. Avaliação dos subconjuntos encontrados

Os algoritmos de busca utilizados na primeira etapa podem ser divididos em 3 grupos principais: algoritmos exponenciais, seqüenciais ou randômicos [Boz 2002].

Algoritmos exponenciais, como a busca exaustiva, fazem todas as combinações de atributos possíveis antes de retornar um subconjunto de atributos. Eles são normalmente inviáveis computacionalmente, por seu tempo de execução crescer exponencialmente com o número de atributos disponíveis. Exemplos clássicos de métodos que utilizam esses conceitos são a busca em largura e a busca em profundidade [Liu e Motoda 1998].

Algoritmos seqüenciais, como a seleção seqüencial para frente e a seleção seqüencial pra trás, podem ser muito eficientes, na resolução de muitos problemas de seleção de atributos, mas têm a desvantagem de não trabalhar com interação entre atributos.

A seleção seqüencial pra frente inicia a busca pelo melhor subconjunto de atributos com um conjunto vazio de atributos. Inicialmente, subconjuntos de atributos com apenas um atributo são avaliados, e o melhor atributo A^* é selecionado. Esse atributo A^* é então combinado com todos os atributos disponíveis (em pares), e o melhor subconjunto de atributos é selecionado. A busca continua dessa mesma forma, sempre adicionando um atributo por vez ao melhor subconjunto de atributos anteriormente selecionado, até que não se consiga mais melhorar a qualidade do subconjunto de atributos selecionados.

A seleção seqüencial para trás, ao contrário da seleção seqüencial para frente, inicia a busca por um subconjunto de atributos ótimos com uma solução representando todos os atributos, e a cada iteração um atributo é removido da solução atual, até que não se consiga melhorar a qualidade da solução encontrada.

Algoritmos genéticos são um ótimo exemplo de métodos de busca randômicos, e sua principal vantagem sobre métodos seqüenciais é justamente tratar do problema de interação entre atributos [Freitas 2001].

Um exemplo de como o problema da interação entre atributos pode enganar um algoritmo de busca guloso é mostrado na Figura 2.3, onde os atributos A_1 e A_2 são utilizados para prever um atributo meta B. O atributo B é determinado de acordo com a função lógica XOR, que assume valor 1 apenas se A_1 e A_2 assumirem valores diferentes.

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

Figura 2.3: Exemplo de interação entre atributos em problema de classificação do tipo XOR

Suponhamos que um método guloso, como o C4.5, que considera apenas um atributo por vez, tente encontrar um modelo para a função acima. Se a condição ($A_1 = 0$) é selecionada para tentar iniciar a construção da árvore, ela não será útil, uma vez que, considerando todos os dados, a distribuição de classes é igual (50% de exemplo com $B = 0$ e 50% com $B = 1$). O mesmo acontecerá se qualquer outro par atributo/valor for selecionado, pois a distribuição das classes permanece constante.

Assim, o C4.5 concluiria que nenhum dos atributos previsores é relevante para determinar o atributo meta. Porém, essa conclusão não é verdadeira, pois precisamos conhecer o valor dos dois atributos para prever o terceiro.

O simples exemplo mencionado na Figura 2.3 mostra que, para trabalhar com interação entre atributos, precisamos de métodos que consideram vários atributos ao mesmo tempo, durante o processo de indução.

Em relação ao tipo de avaliação que pretendem realizar, algoritmos de busca podem implementar dois tipos de abordagem: a abordagem filtro ou a *wrapper*. Essas abordagens independem do método ou algoritmo utilizado na seleção, e são caracterizadas por seu grau de dependência em relação ao algoritmo de classificação.

A abordagem *wrapper* define um subconjunto ótimo de soluções de acordo com uma base de dados e algoritmo de indução particulares, levando em conta a tendência (*bias)* indutiva do algoritmo e sua interação com o conjunto de treinamento. A Figura 2.4 esquematiza um algoritmo de seleção de atributos utilizando a abordagem *wrapper*.

A abordagem filtro, ao contrário da *wrapper*, tenta escolher um subconjunto de atributos independente do algoritmo de classificação, estimando a qualidade dos atributos apenas em relação aos dados. A Figura 2.5 mostra a seleção de atributos com a abordagem filtro, que faz a seleção usando uma etapa de pré-processamento, baseada nos dados de treinamento. Durante essa fase, os conjuntos de atributos gerados são avaliados de acordo com alguma heurística simples, como a ortogonalidade dos dados [Bala *et* al 1996].

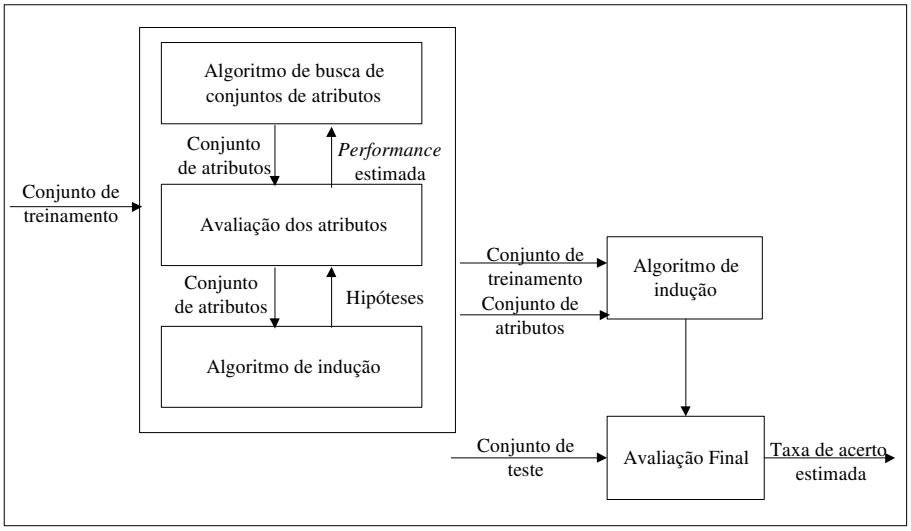


Figura 2.4: Seleção de Atributos utilizando abordagem *wrapper* [Kohavi e John 1998]

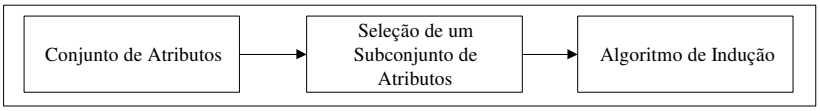


Figura 2.5: Seleção de Atributos utilizando abordagem Filtro

A abordagem *wrapper* normalmente aumenta consideravelmente o tempo de execução do algoritmo, mas a precisão preditiva obtida tende a ser superior àquela obtida pela abordagem filtro.

2.3 Critérios para avaliar a qualidade do conhecimento descoberto

Medir a qualidade do conhecimento extraído através de um algoritmo de mineração de dados não é tarefa simples, uma vez que múltiplos objetivos devem ser considerados durante a avaliação, sendo alguns deles muito subjetivos [Freitas 2002].

Normalmente procuramos conhecimento correto, compreensível e interessante. Essas três qualidades devem estar presentes e devem ser analisadas separadamente, para evitar que o conhecimento descoberto seja similar a

Se (grávida = sim) então (sexo = feminino).

O conhecimento expresso pela regra acima é correto, simples, mas nada surpreendente.

Dessa forma, quando a tarefa alvo do algoritmo de mineração de dados é a classificação, o conhecimento descoberto é considerado correto se a taxa de acerto do classificador em dados de teste for alta.

A simplicidade do conhecimento é subjetiva, uma vez que o que é simples para um usuário pode não ser para outro. Porém, no caso de classificadores que fornecem como saída regras ou árvores de decisão, na literatura geralmente considera-se que a simplicidade está diretamente relacionada ao número de antecedentes de regra e ao número de regras que um modelo de aprendizagem produz, ou ao tamanho da árvore de decisão gerada por ele. Quanto menor o número de regras (ou o número de condições no antecedente de uma regra), mais simples ela é. O mesmo é válido para árvores de decisão. Quanto menor a árvore, maior sua simplicidade.

O interesse ou surpresa do usuário em relação ao conhecimento descoberto é também considerado um critério subjetivo, e está baseado principalmente nos conhecimentos prévios e expectativas do usuário. Porém, algumas medidas objetivas tentam estimar esse critério, baseando-se apenas nos dados que estão sendo minerados. O ideal é combinar formas de medida subjetivas com medidas objetivas, como feito em [Liu *et* al 1997].

2.4 Otimização Multiobjetiva

Grande parte dos problemas do mundo real envolve a otimização de múltiplos objetivos. Porém, a maioria dos métodos para resolução desses problemas evita as complexidades que um problema (otimização) multiobjetivo envolve. Com isso, surgiram muitos métodos para converter problemas multiobjetivos em problemas com um único objetivo [Deb 2001].

Dispondo-se de vários métodos de conversão, esqueceu-se que, na verdade, o problema de otimização com um objetivo é degenerado de problemas com mais de um objetivo, e que existem diferenças fundamentais entre eles. A principal delas está na solução do problema. Por tratar de objetivos conflitantes, na otimização multiobjetivo cada objetivo corresponde a uma solução ótima. Isso faz com que esses problemas apresentem várias soluções ótimas, enquanto que algoritmos que solucionam problemas de otimização com um objetivo normalmente geram apenas uma solução ótima.

Mesmo levando em conta essa diferença fundamental entre problemas com um ou vários objetivos sabemos que, independente do tipo de problema a ser resolvido, no mundo real, sob o ponto de vista prático, necessitamos de apenas uma solução.

Assim, para problemas de otimização multiobjetiva, cabe ao usuário, utilizando informações não-técnicas e qualitativas, optar por uma das soluções apresentadas como sendo a solução ótima para o problema.

A Figura 2.6 mostra um esquema de um procedimento de otimização multiobjetiva ideal. No Passo 1, múltiplas soluções são encontradas, enquanto no passo 2 uma delas é selecionada de acordo com as necessidades do usuário.

Pela Figura 2.6, observamos que realmente a otimização com apenas um objetivo é degenerada da multiobjetiva. Se o esquema que ela representa tivesse que ser modificado para representar a resolução de um problema com um objetivo, no Passo 1 apenas uma solução seria encontrada, e o Passo 2 não precisaria ser executado.

Cada solução encontrada como sendo ótima para um problema multiobjetivo corresponde a uma ordem específica de importância dos objetivos. Se uma preferência em relação aos objetivos é conhecida, não é necessário aplicar o esquema apresentado na Figura 2.6.

Um método simples pode ser utilizado para criar uma função composta objetiva, definida como a soma dos objetivos com seus respectivos pesos, em que o peso é proporcional ao fator de preferência de um objetivo em particular.

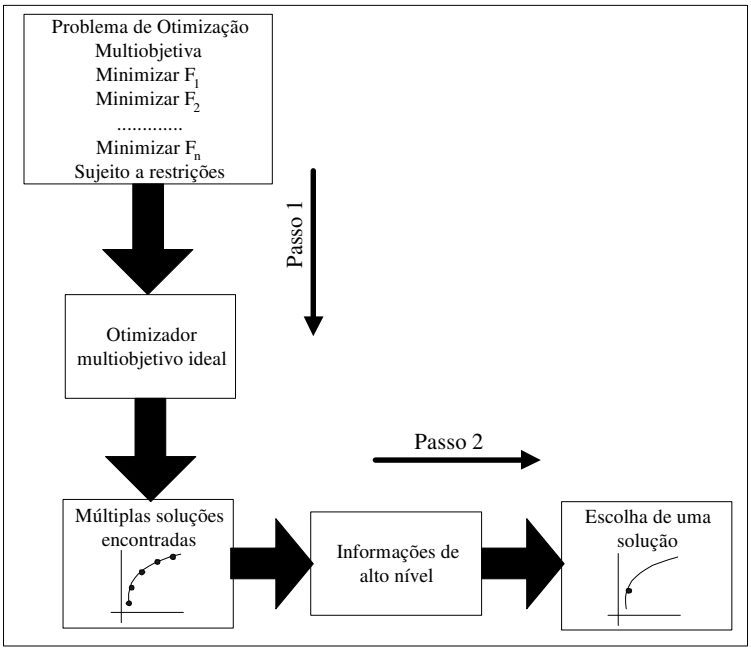


Figura 2.6: Otimização multiobjetiva ideal [Deb 2001]

O método citado acima é denominado otimização multiobjetiva baseada em preferência, e é apenas um dos métodos utilizados para conversão de problemas

multiobjetivos em problemas com um objetivo e, apesar de muito simples, é mais subjetivo que o procedimento ideal mostrado na Figura 2.6. A Figura 2.7 mostra um esquema do método baseado em preferência.

A diferença essencial nesses dois esquemas de otimização é que, no esquema ideal, a informação do problema não é utilizada para buscar por uma nova solução, e sim para escolher uma solução dentre um conjunto de soluções ótimas já escolhidas. Na otimização baseada em preferência, a informação deve ser fornecida antes da busca ser iniciada, sem nenhum conhecimento das possíveis conseqüências.

Podemos então concluir que a abordagem multiobjetiva ideal é mais prática, mais metódica e menos subjetiva que a baseada em preferência. Porém, se um vetor de preferências para o problema já é conhecido, não há razões para não utilizá-lo [Deb 2001].

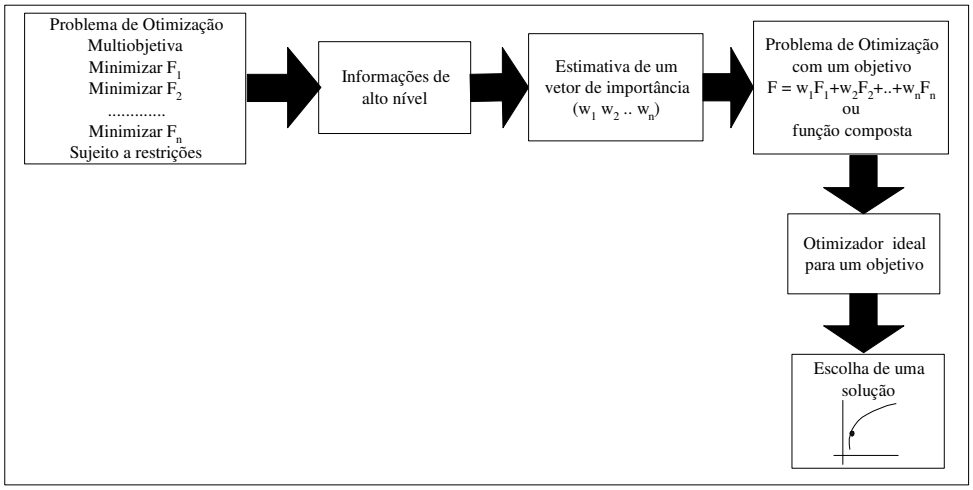


Figura 2.7: Otimização multiobjetiva baseada em preferência [Deb 2001]

Como já comentado, os princípios da otimização multiobjetiva são diferentes dos da otimização com apenas um objetivo. Enquanto a última encontra apenas uma solução global ótima, a primeira, como possui vários objetivos, pode apresentar uma solução ótima diferente para cada objetivo. Se existem diferenças suficientes entre soluções ótimas correspondentes a diferentes objetivos, as funções objetivo são ditas conflitantes entre si. Quando isso ocorre, um conjunto de soluções ótimas surge. A essas soluções ótimas chamamos soluções ótimas de Pareto.

Consideremos um problema com dois objetivos a serem minimizados: o número de acidentes e custo de fabricação de um certo produto. Figura 2.8 mostra um conjunto de possíveis soluções para o problema. Observemos que, por exemplo, a solução A está

próxima do custo mínimo, mas possui uma alta taxa de acidentes. Por outro lado, o ponto B apresenta uma solução custosa, mas que não tem tendência a acidentes. Se os dois objetivos são importantes para a solução do problema, não podemos dizer que a solução A é melhor que a solução B, e vice-versa. Uma solução é melhor que a outra em um objetivo, mas pior em outro. A solução D, que também pertence ao conjunto de soluções ótimas, também não pode ser considerada melhor ou pior que A e B. Assim, todas as soluções que na Figura 6 aparecem como pertencentes ao fronte de Pareto (linha pontilhada), são consideradas soluções ótimas de Pareto.

Observando ainda a Figura 2.8, concluímos que existem também algumas soluções não-ótimas de Pareto, como C. Se compararmos a solução C com A, nós novamente não podemos dizer que A ou C sejam melhores nos dois objetivos. Porém, C não faz parte do fronte de Pareto porque existe uma solução D que é melhor que C nos dois objetivos. Por esse motivo, a solução C é conhecida como uma solução dominada. Nesse exemplo, a solução C também é dominada pela solução B.

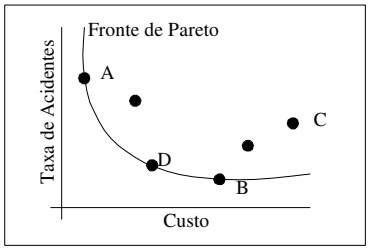


Figura 2.8: Conceito de soluções ótimas de Pareto [Deb 1999]

Dessa forma, podemos concluir que para um problema com mais de uma função objetiva, uma solução x_1 domina uma solução x_2 se duas condições são satisfeitas:

1. A solução x_1 não é pior que x_2 em nenhum dos objetivos.
2. A solução x_1 é estritamente melhor que a solução x_2 em pelo menos um objetivo.

Assim, as soluções que não são dominadas por nenhuma outra solução são consideradas soluções ótimas de Pareto.

Da mesma forma que métodos tradicionais de busca, que trabalham com apenas uma solução, são aplicados para solucionar problemas com um objetivo, a otimização multiobjetiva exige que o método de busca designado para resolver o problema encontre um conjunto de soluções ideais, e não uma solução global ideal.

Dos métodos de busca não convencionais conhecidos hoje, os algoritmos evolucionários destacam-se por utilizar uma população de soluções a cada iteração, e retornar um conjunto de soluções.

"A habilidade dos algoritmos evolucionários em encontrar múltiplas soluções ótimas em uma simulação faz com que sejam únicos resolvendo problemas de otimização multiobjetiva." [Deb 2001].

3. ALGORITMOS GENÉTICOS (AG)

Um algoritmo genético faz implicitamente o que é inviável explicitamente. Foi assim que John Holland, o primeiro idealizador dos algoritmos genéticos, certa vez os definiu. E é por esse mesmo motivo que, nas últimas décadas, algoritmos genéticos vem sendo muito utilizados como métodos de busca e otimização em vários domínios, incluindo mineração de dados. Outras razões para seu sucesso relacionam-se à busca global que eles realizam, à sua facilidade de uso e à vasta aplicabilidade nos mais diferentes domínios [Goldberg 1989].

Algoritmos genéticos são baseados nos mecanismos de seleção natural e envolvem a sobrevivência do indivíduo mais apto. Eles são especialmente atrativos por não exigirem que se saiba como encontrar uma solução ótima para um problema, mas sim como reconhecê-la como ótima.

Sua aplicação na resolução de um problema deve seguir alguns passos. Inicialmente, deve-se criar uma representação da solução do problema através de um indivíduo (cromossomo), e definir como a avaliação dessa solução será realizada (função de *fitness*). Além disso, é preciso determinar como a população inicial será gerada (na maioria dos AGs, sua inicialização é feita de forma aleatória) e como e quais operadores genéticos serão aplicados às soluções. Por último, um conjunto de parâmetros como tamanho da população, número de gerações e probabilidades de aplicação dos operadores deve ser definido [Michalewicz 1996]. (A maioria desses parâmetros tradicionais já possui valores padrão definidos, como mostrado em [Martín-Baurista e Vila 1999], através de comparações entre diversos AGs).

A Figura 3.1 mostra detalhadamente o fluxo de um algoritmo genético simples.

O critério de parada, representado na Figura 3.1 por “Cond?”, é alcançado quando um número de gerações previamente definido é alcançado, quando uma solução suficientemente boa é encontrada ou quando o sistema não consegue mais evoluir [Berson e Smith 1997].

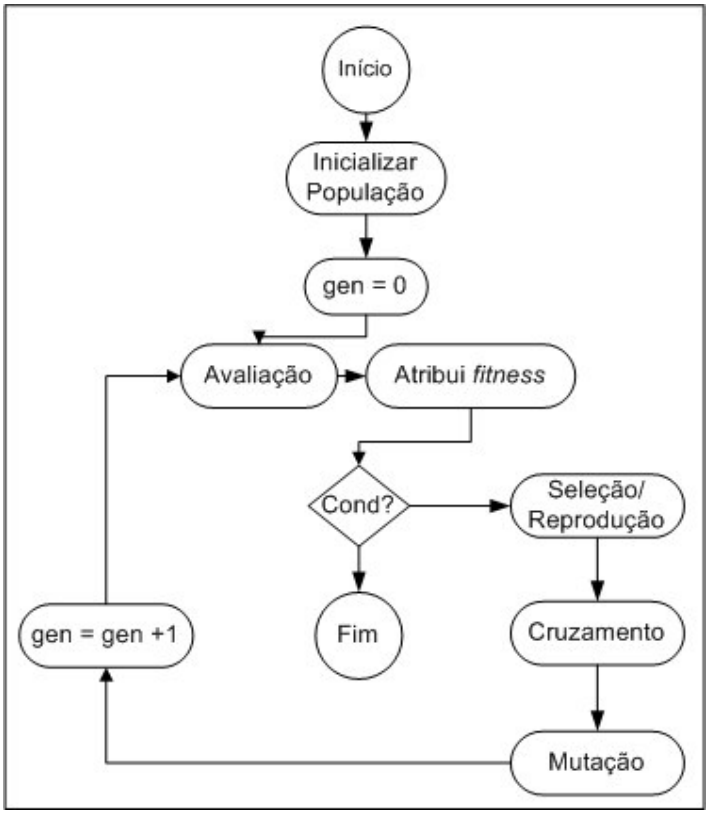


Figura 3.1: Diagrama de fluxo de um AG Simples [Deb 2001]

As próximas seções apresentam com detalhes os componentes de um algoritmo genético.

3.1 Codificação do Indivíduo

Os indivíduos com os quais o AG trabalha representam uma solução candidata à resolução do problema em questão. Encontrar a melhor representação para uma solução candidata de acordo com o problema é sempre desejável.

Existem várias maneiras de representar um indivíduo em um AG. A mais simples e comumente utilizada é a representação binária de tamanho fixo, em que um indivíduo é uma cadeia de bits que assumem valores 0 ou 1 [Hinterding 2000]. Porém, essa representação torna-se problemática quando as variáveis a serem representadas assumem valores contínuos [Freitas 2002].

Em casos em que a representação binária não é a mais natural nem a mais apropriada, outros tipos de representação podem ser utilizados, como sugerido em [Michalewicz 1996].

É interessante ressaltar que, independente do tipo de codificação utilizada, em um algoritmo genético convencional um indivíduo nunca sabe o significado das informações que ele carrega [Dhar e Stein 1997].

3.2 Função de avaliação (*Fitness*)

A função de avaliação é utilizada para determinar o quão boa uma solução candidata é para resolução efetiva de um problema. Somada à forma de codificação do indivíduo, esses dois componentes do AG normalmente são os únicos com relação direta ao domínio do problema.

Em uma população natural, a função de avaliação é determinada pela capacidade do indivíduo de sobreviver a predadores e outros obstáculos naturais, e depois se reproduzir. Em uma população artificial, a responsável pela vida ou morte do indivíduo é sua função objetivo.

Consideremos um exemplo clássico de um AG simples dado por Goldberg [Goldberg 1989], onde queremos encontrar o valor de x que resulte em um valor máximo da função x^2 no intervalo [0,31]. Podemos codificar indivíduos para a solução desse problema com 5 bits, que representam o valor binário de qualquer número no intervalo [0,31].

Nesse caso, poderíamos considerar a função de avaliação como sendo próprio x^2 (função objetivo), visto que quanto maior o valor dessa função, melhor a qualidade de um indivíduo.

A Tabela 3.1 traz exemplos de soluções candidatas para o problema proposto por Goldberg. Dentre as soluções apresentadas, a solução 2 apresenta função de avaliação maior, e por isso pode ser considerada, entre as soluções encontradas, a melhor solução para o problema.

Tabela 3.1: Soluções candidatas para o problema de x^2

Código	**Indivíduo**	**x**	**x^2**
1	01101	13	169
2	11000	24	576
3	01000	8	64
4	10011	19	361

3.3 Métodos de Seleção

Uma vez que AGs baseiam-se no princípio da seleção natural, eles devem ser capazes de identificar os indivíduos mais aptos, para que permaneçam na população durante o processo de evolução, e os mais fracos, para que sejam excluídos do processo.

Vários métodos podem ser utilizados para execução dessa tarefa, entre eles, a seleção proporcional, a seleção por *ranking* e a seleção por torneio [Deb 2001] [Freitas 2002].

Na seleção proporcional, indivíduos são copiados para a próxima geração de acordo com probabilidades proporcionais ao seu valor de função de avaliação. A implementação desse método é normalmente realizada através de um mecanismo de roleta, na qual a roleta é dividida em N partes, N correspondendo ao número de indivíduos da população, e o tamanho de cada uma das partes é proporcional à função de avaliação do indivíduo que representa. A roleta é então girada N vezes, e a cada uma delas o indivíduo indicado pelo ponteiro é selecionado e inserido na nova população.

A seleção por *ranking* pode ser dividida em duas etapas. Na primeira, as soluções são ordenadas de acordo com seus valores da função de avaliação, em ordem crescente, se o propósito for maximizar a função de avaliação, ou em ordem decrescente, caso o objetivo seja minimizá-la.

Estando a lista ordenada, a cada indivíduo é atribuído um novo valor da função de avaliação equivalente a sua posição no *ranking*.

Numa segunda fase, um procedimento similar à seleção proporcional é aplicado. Quanto melhor a posição do indivíduo no *ranking*, maior sua chance de ser selecionado.

Já a seleção por torneio não atribui explicitamente probabilidades aos indivíduos. Através dela, $k \geq 2$ indivíduos, onde k é denominado tamanho do torneio, são escolhidos aleatoriamente a partir da população atual, e suas funções de avaliação comparadas. O indivíduo com melhor valor da função de avaliação é então selecionado para reprodução.

O valor de k é definido pelo usuário. Quanto maior o valor de k, maior a pressão seletiva, ou seja, maior a velocidade com que os indivíduos mais fortes dominam a população, causando a extinção dos mais fracos.

3.4 Elitismo

Visando preservar e utilizar as melhores soluções encontradas na geração atual nas próximas gerações, surgiu a estratégia de elitismo. Em sua versão mais simples, ela conserva os N_{elit} melhores indivíduos da população atual, copiando-os para a próxima geração sem nenhuma alteração. Os outros $N - N_{elit}$ indivíduos da população são gerados normalmente, através do método de seleção e posterior aplicação dos operadores genéticos.

Assim, as melhores soluções não são apenas passadas de uma geração para outra, mas também participam da criação dos novos membros da nova geração.

3.5 Operadores Genéticos

Algoritmos genéticos tradicionais são normalmente constituídos de dois operadores: cruzamento e mutação.

O operador de cruzamento permite a troca de material genético entre dois indivíduos denominados pais, combinando informações de maneira que exista uma probabilidade razoável dos novos indivíduos produzidos serem melhores que seus pais [Hinterding 2000].

A forma mais simples de cruzamento é o cruzamento de um ponto. De acordo com esse método, um ponto do cromossomo é sorteado aleatoriamente, e a troca de material genético feita na região a direita do ponto escolhido. Um exemplo de cruzamento de um ponto pode ser visto na Figura 3.2. A Figura 3.2(a) mostra os dois indivíduos selecionados, e o ponto escolhido é representado pelo símbolo 'l'. A Figura 3.2(b) apresenta os novos indivíduos gerados após o cruzamento.

$X_1\ X_2 \mid X_3\ X_4\ X_5\ X_6$	$X_1\ X_2 \mid Y_3\ Y_4\ Y_5\ Y_6$
$Y_1\ Y_2 \mid Y_3\ Y_4\ Y_5\ Y_6$	$Y_1\ Y_2 \mid X_3\ X_4\ X_5\ X_6$
(a) Antes do Cruzamento	(b) Depois do Cruzamento

Figura 3.2 Exemplo de cruzamento de um ponto [Freitas 2002]

Outra forma de cruzamento muito utilizada é o cruzamento uniforme [Falkenauer 1999]. De acordo com esse método, cada gene do cromossomo (indivíduo) pode ser trocado de acordo com uma probabilidade fixa *p*. Quanto maior o valor de *p*,

maior o número de genes trocados entre dois pais. Na literatura, o valor padrão de *p* é 0.5. A Figura 3.3 mostra um exemplo de cruzamento uniforme. Os genes trocados estão destacados em negrito.

$X_1\ \mathbf{X_2}\ X_3\ \mathbf{X_4}\ \mathbf{X_5}\ X_6$	$X_1\ \mathbf{Y_2}\ X_3\ \mathbf{Y_4}\ \mathbf{Y_5}\ X_6$
$Y_1\ \mathbf{Y_2}\ Y_3\ \mathbf{Y_4}\ \mathbf{Y_5}\ Y_6$	$Y_1\ \mathbf{X_2}\ Y_3\ \mathbf{X_4}\ \mathbf{X_5}\ Y_6$
(a) Antes do Cruzamento	(b) Depois do Cruzamento

Figura 3.3 Exemplo de cruzamento uniforme [Freitas 2002]

A mutação tem como objetivo substituir o valor de um gene por um outro valor aleatoriamente gerado. No caso do indivíduo ser representado por um *string* binário, ela consiste em escolher aleatoriamente um gene do cromossomo e inverter seu valor de 1 para 0 ou vice-versa.

O propósito da mutação é manter a diversidade da população e assegurar que o cromossomo sempre cobrirá uma parte suficientemente grande do espaço de busca [Hinterding 2000]. Ela é normalmente aplicada em baixas freqüências, pois freqüências muito altas implicariam busca aleatória.

3.6 Niching

A técnica de *niching* consiste na divisão da população em espécies (que reúnem indivíduos com características semelhantes) para reduzir a competição por recursos e criar subpopulações estáveis, cada uma delas concentrada em um nicho do espaço de busca.

Métodos de *niching* são conhecidos por sua capacidade de criar e manter populações diversas. Dois tipos de método são normalmente utilizados na literatura: *sharing* e *crowding*.

O mecanismo de *sharing* altera apenas o procedimento de atribuição do valor da função de avaliação de um indivíduo. Ele trabalha alterando a função de avaliação de cada elemento da população de acordo com o número de indivíduos semelhantes a ele na população [Mathfound 2000].

O compartilhamento da função de avaliação (*fitness sharing*) de um indivíduo, denominado F', é igual a sua função de avaliação F dividida por seu contador de nichos

(*niching count*). O contador de nichos é a soma dos valores das funções de compartilhamento (*sh*) entre ele e os demais indivíduos da população (incluindo ele mesmo). A Fórmula 3.1 define formalmente o compartilhamento de função de avaliação de um indivíduo *i*, onde *u* é o número de indivíduos da população.

$$F'(i) = \frac{F(i)}{\sum_{j=1}^{u} sh(d(i,j))} \qquad (3.1)$$

A função de compartilhamento é gerada em função de uma distância *d* entre dois elementos da população, e retorna 1 se os elementos são iguais, 0 se a diferença entre eles é maior que um limiar de dissimilaridade, e um valor intermediário entre 0 e 1 de acordo com seus níveis de dissimilaridade. Um limiar de dissimilaridade é especificado por uma constante denominada $\sigma_{share.}$ Se a distância entre dois elementos da população for maior ou igual a $\sigma_{share,}$ eles não afetam o compartilhamento da função de avaliação um do outro. Assim,

$$sh(d) = \begin{cases} 1-(d-\sigma_{share})^{\alpha}, se \quad d < \sigma_{share} \\ 0, \textit{caso contrário} \end{cases} \qquad (3.2)$$

onde α é uma constante que regula a forma do compartilhamento da função de avaliação, e tem valor padrão na literatura igual a 1.

Sharing pode ser utilizado considerando tanto distâncias de fenótipo quanto de genótipo.

Já o método de *crowding* insere novos indivíduos na população substituindo indivíduos similares. Da mesma forma que o método de *sharing*, ele utiliza uma medida de distância, genotípica ou fenotípica, para encontrar indivíduos similares [Deb e Goldberg 1989].

3.7 Algoritmos Genéticos Multiobjetivos (AGMO)

Não é novidade o sucesso do uso de algoritmos genéticos em problemas de otimização, incluindo a otimização multiobjetiva. Além das vantagens que os AGs oferecem na resolução de qualquer problema de busca ou otimização, citadas no início desse Capítulo, em relação à otimização multiobjetiva eles apresentam uma vantagem a

mais. Como a otimização multiobjetiva busca a otimização de objetivos conflitantes, em que cada um deles corresponde a uma solução ótima, isso faz com que esses problemas apresentem várias soluções ótimas, e sejam preferencialmente resolvidos por métodos capazes de gerar um conjunto de soluções ótimas de uma só vez. Nesse caso, AGs seriam recomendados.

Utilizando os conceitos básicos de AGs, definidos nas seções anteriores, Schaffer, em 1984, realizou a primeira implementação de um algoritmo genético multiobjetivo, denominado VEGA (*Vector- Evaluated Genetic Algorithm*). A primeira idéia de utilizar um método de busca genética para encontrar a solução de um problema com mais de um objetivo surgiu bem antes, em 1967, com Rosenberg, mas só veio a ser implementada por Schaffer.

Schaffer modificou um algoritmo genético simples para que executasse ciclos independentes de seleção de acordo com cada objetivo. Ou seja, considerando um AG com população de *N* indivíduos e *k* objetivos a serem otimizados, *k* sub-populações com *N / k* indivíduos são geradas. Essas populações são posteriormente unidas e os operadores de cruzamento e mutação aplicados. A Figura 3.4 mostra o esquema acima descrito.

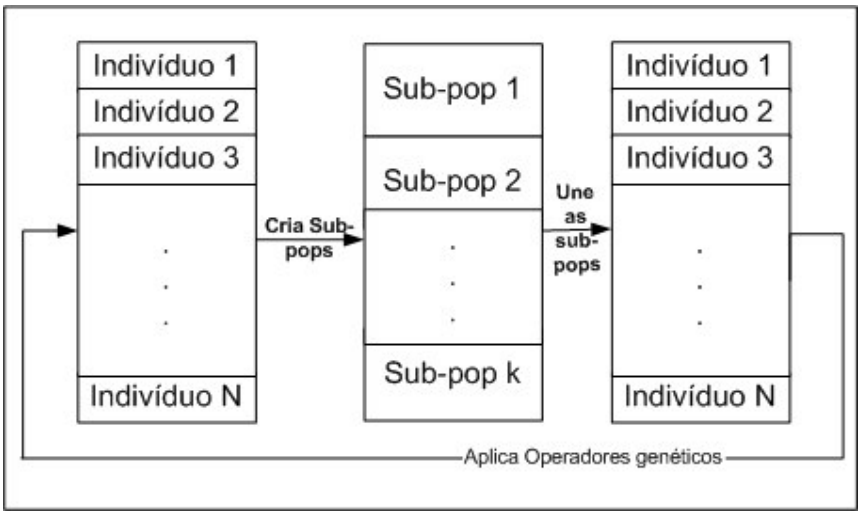


Figura 3.4: Mecanismo de Seleção do VEGA [Coello *et* al 2002]

Depois desse primeiro trabalho de Schaffer, nenhum outro foi realizado até que Goldberg [Goldberg 1989] esboçasse em 10 linhas o que ele chamou de um procedimento de ordenação (*ranking*) de indivíduos não dominados. Sua sugestão era

utilizar o conceito de dominância para deixar várias cópias de indivíduos não dominados em uma população. Para que essas cópias não afetassem a diversidade da população, ele ainda sugeriu o uso de *niching* entre as soluções de uma classe de soluções não dominadas.

Partindo dessa sugestão de Goldberg, vários pesquisadores desenvolveram diferentes versões de algoritmos genéticos multiobjetivos [Coello *et* al 2002], [Deb 2001]. Entre elas, destacam-se o MOGA (*Multiobjective Genetic Algorithm*) [Fonseca e Fleming 1993], o NSGA (Non-Dominated Sorting Genetic Algorithm) [Srinivas e Deb 1994], o NPGA (*Niched Pareto Genetic Algorithm*) [Horn *et* al 1994] e o SPEA (*Strength Pareto genetic Algorithm*) [Zitzler e Thiele 1999].

A principal diferença entre esses algoritmos consiste na forma em que o valor da função de avaliação é atribuído aos indivíduos da população. Esse procedimento pode ser baseado em uma abordagem agregada (que utiliza métodos que atribuem pesos a cada objetivo e os unem em uma só função, como o HLGA (Hajela's and Lin's Weighting-based Genetic Algorithm)), abordagens não Pareto (como a utilizada pelo VEGA) e abordagens baseadas em Pareto (como o NSGA e NPGA) [Fonseca e Fleming 1994].

As próximas seções descrevem resumidamente o funcionamento do MOGA, NSGA, NPGA e do SPEA. Os três primeiros merecem atenção por serem métodos tradicionais da literatura, servindo de base para o desenvolvimento de muitos outros algoritmos genéticos multiobjetivos. Já o SPEA é um método mais atual, e une em um único algoritmo as potencialidades dos métodos multiobjetivos tradicionais. Ele é hoje objeto de pesquisa em muitas áreas, principalmente por sua superioridade em relação a outros métodos multiobjetivos, e foi utilizado como método de comparação nos experimentos realizados neste trabalho.

3.7.1 MOGA- *Multiobjective Genetic Algorithm*

Fonseca e Fleming foram os primeiros a sugerir um AGMO que explicitamente enfatiza soluções não dominadas e ao mesmo tempo mantém a diversidade entre elas. Seu funcionamento pode ser resumido em 3 etapas:

1. Ordena toda a população de acordo com diferentes classes de indivíduos não dominados. Inicialmente, o algoritmo encontra todos os indivíduos não

dominados da população e insere-os na Classe 1. Dos indivíduos remanescentes na população (número de indivíduos da população – número de indivíduos na classe 1), os não dominados são novamente selecionados, e inseridos na Classe 2. Esse processo continua até que não existam mais indivíduos na população.

2. Indivíduos pertencentes a classe 1 (classe com os melhores indivíduos) recebem *rank* 1. O *rank* dos outros indivíduos é atribuído de acordo com o número de soluções *k* que dominam esse indivíduos, acrescido de um. Consideremos por exemplo uma solução S_1, pertencente a classe 2, e dominada por 3 outros indivíduos da população. Essa solução S_1 recebe *rank* 4.

3. Finalizado o processo de ordenação, uma *raw fitness* (função de avaliação "bruta") é atribuída a cada solução de acordo com seu *rank*. Para isso, os *ranks* são ordenados de forma crescente, e uma função de mapeamento linear (ou de um outro tipo qualquer) é utilizada para atribuir uma *raw fitness* a cada solução. Normalmente, a função de mapeamento é escolhida de forma que os valores de *fitness* atribuídos variem entre *N* (*raw fitness* da melhor solução do *ranking*) e 1 (*raw fitness* da pior solução do ranking). Por último, é calculada uma média das *raw fitness* das soluções de um mesmo *rank.* O procedimento de seleção usa então essas *raw fitness*, que passam a ser denominadas funções de avaliação atribuídas, para selecionar ou apagar blocos de soluções. Durante a seleção, o procedimento de *niching* é executado, para distribuir a população ao longo da região ótima de Pareto.

3.7.2 NSGA (*Non-Dominated Sorting Genetic Algorithm*)

O NSGA é um método similar ao MOGA. Suas principais diferenças encontram-se na maneira como a função de avaliação é atribuída ao indivíduo e na estratégia de *niching*.

A execução do NSGA inicia-se com a busca da melhor classe de indivíduos não dominados da população. Terminado esse processo, a cada indivíduo é atribuído um valor de função de avaliação igual a *N*, onde *N* é o número de indivíduos da população.

Uma estratégia de *sharing* é então utilizada para encontrar o *niche count* de cada indivíduo da melhor classe de não dominados. A cada indivíduo uma *shared fitness* é

atribuída dividindo o valor anteriormente atribuído a *fitness* (igual ao número de indivíduos da população) pelo *niche count* encontrado.

O menor valor de *shared fitness* atribuído é guardado em Sh_{min}. Posteriormente, uma segunda melhor classe de não dominados é encontrada, e a cada indivíduo é atribuído um valor de função de avaliação igual a Sh_{min}- v, onde v é um número positivo pequeno. *Niche count* e *shared fitness* são então calculados. Esse processo é repetido até que as funções de avaliação de todos os indivíduos tenham sido encontradas.

Esse procedimento de atribuição de função de avaliação garante que:

- A uma solução dominada é sempre atribuído um valor de *shared fitness* menor que o de qualquer solução que a domine.
- Para cada classe de não dominados, a diversidade seja garantida.

3.7.3 NPGA (*Niched Pareto Genetic Algorithm*)

O NPGA difere dos algoritmos descritos nas seções anteriores em seu esquema de seleção, baseado em torneios de dominância de Pareto. Para isso, um conjunto de comparação *C,* composto por um número específico de indivíduos representado por t_{dom} é retirado aleatoriamente da população, no início da seleção.

Quando a seleção por torneio é iniciada, 2 indivíduos são aleatoriamente escolhidos da população, e o vencedor é determinado de acordo com suas relações de dominância considerando o subconjunto *C*. Assim, se um dos indivíduos que concorrem ao torneio dominar todos os t_{dom} indivíduos de *C,* ele é declarado vencedor. Caso contrário, um *niche count* é calculado para cada indivíduo, levando em conta toda a população. A solução com o menor *niche count* vence o torneio.

3.7.4 SPEA (*Strength Pareto Evolutionary Algorithm*)

O conceito do SPEA une todas as potencialidades de diversos algoritmos em um único. Ele é caracterizado por armazenar em um conjunto externo *P'* todas as soluções não dominadas encontradas da primeira à última geração do algoritmo. No caso desse conjunto exceder um número máximo de indivíduos *max*, um algoritmo de agrupamento é utilizado para reduzir o tamanho do conjunto.

Quanto ao mecanismo de atribuição de função de avaliação, o algoritmo trabalha em 2 estágios.

1. Ordenação dos elementos do conjunto *P'*: A cada solução *i* pertencente a *P'*, um valor real s_i entre [0,1), denominado *strength*, é atribuído. O valor de s_i é proporcional ao número de indivíduos da população atual *P* que são dominados pelo indivíduo *i*. Assim, sendo *n* o número de indivíduos de *P* que são dominados por *i* e *N* o número de indivíduos na população *P*, $si = \frac{n}{N+1}$. A função de avaliação de cada indivíduo *i* é equivalente a seu s_i.

2. A função de avaliação de cada indivíduo *j* pertencente a *P* é equivalente a soma de todas as *strengths* de todos os indivíduos *i* de *P'* que dominam *j* somado de 1.

A maneira como a função de avaliação é atribuída a cada um dos indivíduos e seus efeitos podem ser claramente visualizados na Figura 3.5, que representa um problema de maximização dos valores das funções f_1 e f_2.

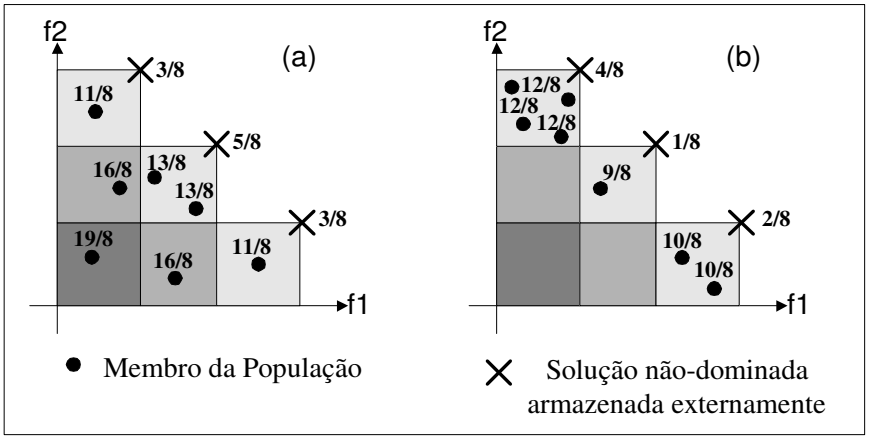


Figura 3.5: Dois cenários de um problema de maximização de dois objetivos e o comportamento do SPEA [Zitzler e Thiele 1999]

Na Figura 3.5, o espaço de objetivos, coberto por três soluções não dominadas, é dividido em três áreas distintas, representadas por retângulos. Cada subconjunto de *P'* define uma área dominada por todos seus elementos. O retângulo mais escuro, no canto inferior esquerdo do gráfico, agrupa indivíduos dominados por todas as soluções não dominadas encontradas, enquanto a área mais clara, no canto superior esquerdo é dominada apenas por uma dessas soluções.

Cada uma dessas áreas é considerada pelo algoritmo como um nicho, e o objetivo do método é distribuir os indivíduos através de todas essas áreas de forma que:

1. As áreas mais claras, dominadas por poucas soluções, contenham mais indivíduos que as áreas escuras, dominadas por mais soluções.
2. Uma área englobe tantos indivíduos quanto outra área dominada pelo mesmo número de soluções não dominadas.

Esse mecanismo seleciona intuitivamente soluções próximas ao fronte de Pareto, distribuindo-as em torno de sua superfície.

A Figura 3.5 (a) ilustra o primeiro dos objetivos a serem alcançados pelo método: as funções de avaliação dos indivíduos nas áreas mais claras possuem valores superiores ao do restante da população. A Figura 3.5 (b) permite a visualização do princípio da força (*strength*): indivíduos com mais vizinhos em seu nicho são penalizados devido ao alto valor de força da solução não dominada a eles associada. Quanto mais forte a solução não dominada, menores os valores das funções de avaliação dos indivíduos por ela dominados.

Segundo [Coello *et* al 2002], o SPEA pertence ao que ele considera a segunda geração de técnicas para desenvolvimento de algoritmos multiobjetivos. A segunda geração é caracterizada por primar pela eficácia dos métodos, em que é comum a presença de populações externas e a busca por soluções não dominadas e, ao mesmo tempo, bem distribuídas por todo o fronte de Pareto.

Um estudo realizado por Zitzler [Zitzler *et* al 2000] compara o desempenho de alguns algoritmos genéticos multiobjetivos, entre eles, VEGA, MOGA, NPGA, NSGA e SPEA. Algumas métricas são utilizadas para comparar as soluções encontradas pelo algoritmo, sempre levando em conta a distância entre o fronte de Pareto encontrado e o fronte de Pareto ideal, a distribuição das soluções no fronte de Pareto e o tamanho do intervalo de valores que cada um dos objetivos é capaz de cobrir.

De acordo com essas métricas, os algoritmos foram testados em 6 funções de teste, e uma hierarquia entre eles foi encontrada. O SPEA foi considerado o mais eficaz deles, seguido pelo NSGA e VEGA.

Uma revisão mais completa sobre algoritmos genéticos multiobjetivos pode ser encontrada em [Deb 2001] e [Coello *et* al 2002].

4. MÉTODO PROPOSTO

Algoritmos genéticos vêm sendo muito utilizados na resolução de problemas de seleção de atributos, principalmente por sua aplicação ser vantajosa em situações em que o espaço de busca é grande. Tratando-se de seleção de atributos, a dimensão do espaço de busca está diretamente relacionada ao número M de atributos contidos na base de dados, e é definida como 2^M.

Além disso, AGs realizam um processo de busca global através da população de indivíduos, enquanto nos algoritmos de busca tradicionais esse processo é local [Goldberg 1989], [Fidelis *et* al 2000]. Essa busca global permite que interações entre atributos sejam consideradas [Freitas 2001], ponto essencial para um algoritmo de seleção de atributos. Finalmente, conforme mencionado na Seção 3.6, AGs são métodos naturais para implementação de buscas multiobjetivo, em que se deseja encontrar múltiplas soluções ótimas, pois AGs trabalham com populações de soluções.

Considerando essas vantagens, esse trabalho propõe um algoritmo genético multiobjetivo (AGMO) para seleção de atributos.

Dado um conjunto de atributos *A*, o algoritmo tem como objetivo encontrar um subconjunto de atributos *A'* que reduza tanto a taxa de erro de classificação, quanto a quantidade de regras produzidas por um algoritmo de classificação qualquer, melhorando assim tanto o poder preditivo quanto a compreensibilidade da saída do classificador.

O cálculo da função de avaliação (*fitness)* é baseado na abordagem *wrapper*, que avalia a qualidade dos atributos selecionados, utilizando o próprio algoritmo de classificação alvo.

No trabalho proposto, o algoritmo de mineração de dados utilizado é o C4.5 [Quinlan 1993], e a qualidade do subconjunto de atributos selecionados é determinada pela taxa de erro de classificação e pelo tamanho da árvore obtidos, durante a fase de treinamento do C4.5.

Além do AGMO, este trabalho propõe também uma versão multiobjetiva de um dos algoritmos mais conhecidos na área de seleção de atributos: o algoritmo de seleção seqüencial para frente.

O algoritmo de seleção seqüencial para frente multiobjetivo (SSFMO) busca otimizar os mesmos objetivos que o AGMO proposto: a taxa de erro e o tamanho da árvore gerados pelo C4.5, e será utilizado para comparar um método de busca genético com um método de busca guloso, ambos considerando conceitos multiobjetivos na tarefa de seleção de atributos.

A Seção 4.1 traz a descrição completa do AGMO proposto, e a Seção 4.2 destaca os principais aspectos do SSFMO.

4.1 Algoritmo Genético Multiobjetivo para Seleção de Atributos

O AGMO aqui proposto tem como intuito avaliar os efeitos da aplicação dos conceitos de otimização multiobjetiva na área de mineração de dados, mais especificamente seleção de atributos na tarefa de classificação. Portanto, seus fundamentos estão baseados em algoritmos genéticos multiobjetivos já existentes. Suas principais características estão descritas nas próximas seções.

4.1.1 Codificação do Indivíduo

No AG proposto, cada indivíduo representa um subconjunto de atributos selecionados. Cada cromossomo é composto por M genes, onde M é o número de atributos previsores nos dados que estão sendo pré-processados. Cada gene pode assumir valores 0 ou 1, indicando respectivamente a ausência ou presença do atributo no subconjunto criado.

Consideremos, por exemplo, o conjunto de atributos da base *balance-scale* [Murphy e Aha 1994] composta por quatro atributos: peso esquerdo, distância esquerda, peso direito e distância direita. O indivíduo representado na Figura 4.1 seleciona os atributos distância esquerda e peso direito como relevantes e descarta peso esquerda e distância direita.

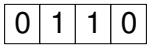


Figura 4.1: Indivíduo representando atributos selecionados na base *balance-scale*

4.1.2 Função de Avaliação (*Fitness*)

A função de avaliação analisa a qualidade de um subconjunto criado de atributos. Com a utilização da otimização multiobjetiva, cada indivíduo *Ii* deve ser avaliado de acordo com dois critérios: pela taxa de erro de classificação do C4.5 quando treinado com os atributos selecionados por *Ii* e pelo tamanho (número de nós) da árvore gerada pelo C4.5.

A Figura 4.2 mostra como é realizado o cálculo da função de avaliação no algoritmo proposto. Inicialmente, os dados de treinamento e validação são preparados, para que apenas os atributos selecionados pelo AG estejam na base de dados. A partição de "teste" é aqui denominada partição de validação, porque o teste propriamente dito é realizado apenas no fim da execução do AG, quando um conjunto de soluções já foi selecionado a partir dos dados de treinamento e validação (ver Seção 4.1.4).

Estando os dados preparados, o C4.5 gera uma árvore de decisão a partir do conjunto de treinamento, cujo tamanho será utilizado como parte da função de avaliação para o indivíduo. Através do conjunto de validação, uma taxa de erro de classificação é também obtida, e armazenada para posterior uso durante a avaliação do indivíduo.

O AG não possui um valor escalar que represente o quão bom um indivíduo é. Quando dois indivíduos são comparados, sua relação de dominância de Pareto é estudada, considerando os dois objetivos a serem otimizados. Se um dos indivíduos comparado domina o outro, ele é considerado como sendo o melhor. Porém, se não existir relação de dominância, um terceiro critério é utilizado para determinar o melhor dos indivíduos. Esse terceiro critério, denominado F_3, é a diferença entre o número de indivíduos que *Ii* domina e o número de indivíduos que dominam *Ii*. F_3 tem sempre valor maior ou igual a zero para os indivíduos não-dominados da população.

O critério F_3 é indispensável na seleção por torneio e na determinação do conjunto elitista.

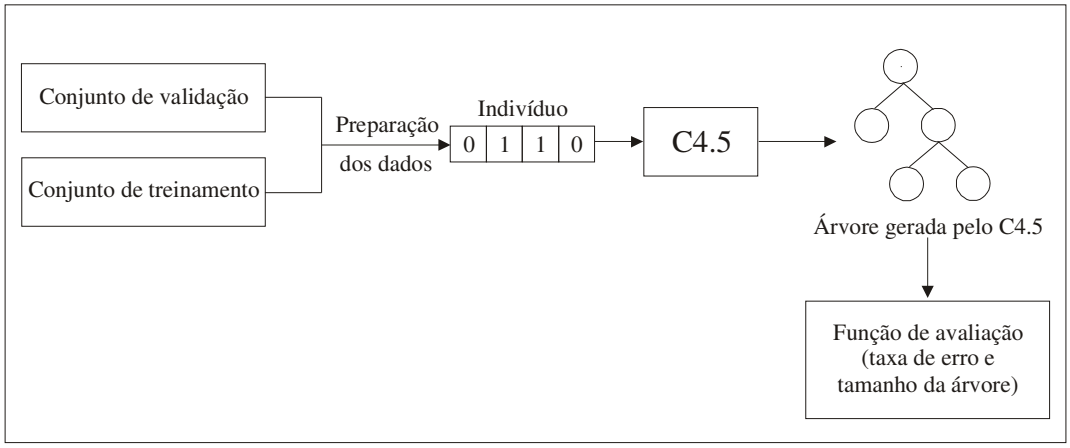


Figura 4.2: Cálculo da função de avaliação

4.1.3 Método de Seleção e Operadores Genéticos

A seleção por torneio é utilizada para determinar os indivíduos aptos e os não aptos a permanecerem na população na próxima geração. Esse método escolhe aleatoriamente *k* indivíduos da população, compara os valores de suas funções de avaliação (*fitness*) e declara vencedor o indivíduo que superar todos os outros. Esse processo é repetido *N* vezes, onde *N* é a diferença entre o número de indivíduos da população e o número de indivíduos selecionados por elitismo. O valor de *k* utilizado pelo algoritmo proposto foi 2.

Como citado na Seção 4.1.2, o AG proposto não tem sua função de avaliação representada por uma medida escalar. Por isso, durante a seleção por torneio, os indivíduos selecionados são comparados de acordo com os conceitos de dominância de Pareto. Se I_1 domina I_2, I_1 vence o torneio, e vice-versa. Caso contrário, F_3 é utilizado, e vence o torneio o indivíduo que apresente maior valor de F_3. São raros os casos em que F_3 não desempata o torneio. Nesse caso, um dos indivíduos é escolhido aleatoriamente como vencedor.

Depois de realizada a seleção por torneio, os indivíduos sofrem processos de cruzamento uniforme e mutação. O cruzamento uniforme foi escolhido por evitar a tendência (*bias)* posicional inserida pelo cruzamento de um ponto, que faz com que a probabilidade de genes vizinhos serem trocados ao mesmo tempo seja muito maior que a de genes distantes serem trocados ao mesmo tempo [Eshelman *et* al 1989]. Esse tipo de tendência não é desejado no caso da seleção de atributos, em que a relevância de um atributo, do ponto de vista de um algoritmo de classificação, é independente de sua

posição dentro do cromossomo. (Do ponto de vista do algoritmo de classificação, há um *conjunto* de atributos, no sentido matemático do termo, em que não há ordenação entre os elementos do conjunto.).

Os operadores de cruzamento e mutação são dependentes de probabilidades pré-definidas pelo usuário. Essas probabilidades foram definidas como 80% para cruzamento e 1% para mutação.

Além dos operadores básicos, um esquema de elitismo foi também incorporado ao método. A função do elitismo é preservar as melhores soluções encontradas, para que essas não se percam durante o processo de evolução. Num AGMO, pode-se considerar que as melhores soluções da geração *i* são aquelas que não são dominadas por nenhuma outra solução da população atual. Assim, o AGMO utiliza um elitismo de não dominados, proposto inicialmente em [Bhattacharyya 2000].

De acordo com esse esquema de elitismo, a cada geração o conjunto de indivíduos não dominados é encontrado, e inicialmente incorporado ao conjunto elitista. Porém, há de se considerar que o número de soluções não dominadas a cada geração é desconhecido e variável, e pode alcançar 100% dos indivíduos da população.

Para evitar a estagnação da população durante o elitismo, o número de indivíduos elitistas está limitado a 50% do número de indivíduos que compõe a população. Seguindo esse esquema, se o número de indivíduos não dominados da população excede o valor máximo, os indivíduos com maior valor de F_3 são preservados. Assim, x indivíduos não dominados na geração i, onde $x \leq$ (tamanho da população)/2, são automaticamente inseridos na geração $i+1$, sem nenhuma alteração.

Segundo estudo feito por [Zitzler *et* al 2000], o elitismo é um fator muito importante em métodos de otimização multiobjetiva, podendo melhorar consideravelmente sua performance.

Além disso, um esquema de *niching* utilizando compartilhamento de função foi adicionado ao método proposto. Porém, ao contrário do esperado, seus resultados não melhoraram significativamente a diversidade das soluções não dominadas encontradas. Dessa forma, como o tempo computacional do algoritmo aumenta muito com a utilização do método de *niching*, ele foi removido do algoritmo proposto.

4.1.4 Solução Retornada

O princípio da otimização multiobjetiva é caracterizado por retornar ao usuário um conjunto de soluções com diferentes *trade-offs* entre os objetivos, e permitir que ele escolha, baseado em informações não técnicas e qualitativas, a solução que lhe for mais conveniente.

Dessa forma, no caso da seleção de atributos, o procedimento ideal para selecionar uma entre as soluções encontradas, analisando os dados de treinamento, seria apresentá-las ao usuário, para que este escolhesse a solução mais conveniente sob seu ponto de vista, para que esta fosse posteriormente analisada nos dados de teste.

Porém, nesta pesquisa, não temos usuários no sentido nobre do termo. Logo, seguir fielmente o princípio de otimização multiobjetiva torna-se impossível. Portanto, na última geração, todas as soluções não dominadas são encontradas e novamente avaliadas num esquema de validação cruzada interna, para que a qualidade das soluções retornadas pelo AG seja legítima.

Um procedimento de validação cruzada de fator k consiste em dividir aleatoriamente as bases de dados utilizadas em experimentos em k partições de mesmo tamanho. Dessas k partições geradas, k-1 são destinadas ao treinamento do algoritmo de classificação e uma para o teste. O procedimento implementado neste trabalho conserva a mesma distribuição de classes para cada uma das partições geradas (validação cruzada estratificada). A Figura 4.3 mostra um esquema de validação cruzada de fator 5. Os dados estão divididos em 5 partições e, a cada iteração, uma delas corresponde ao conjunto de teste (representado pelo quadrado hachurado) e as outras quatro, ao conjunto de treinamento.

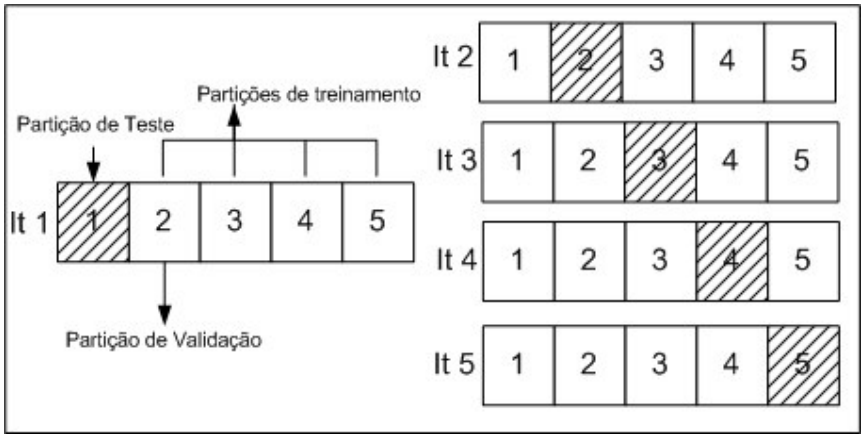


Figura 4.3: Conjuntos de dados na utilização da Validação Cruzada de fator 5

Baseado no conceito descrito acima, o método de validação cruzada interna considera as *k*-1 partições de treinamento como sendo uma base de dados completa, e redivide-as em *k* partições, como mostrado na Figura 4.4.

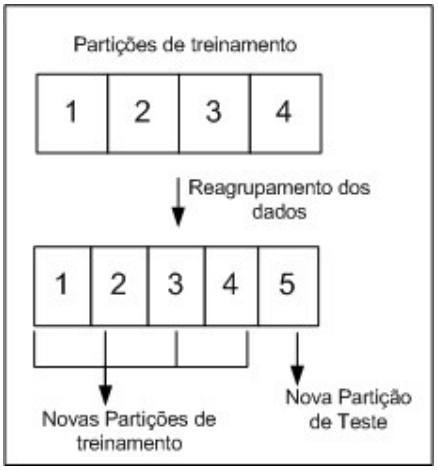


Figura 4.4: Conjunto de dados na Validação Cruzada Interna

No caso do método aqui proposto, a utilização da abordagem *wrapper* exige ainda que um outro tipo de partição de dados, além das de treinamento e teste, seja criada: a partição de validação. Ela é definida como sendo uma das *k*-1 partições de treinamento, e é indispensável na validação dos dados gerados pelo C4.5, antes do teste propriamente dito. Ou seja, durante a avaliação dos indivíduos na evolução do AG, o C4.5 gera uma árvore utilizando as partições de treinamento, e valida a árvore criada utilizando a partição de validação. A partição de teste é utilizada apenas depois que as soluções não dominadas da última geração são encontradas, através do procedimento de validação cruzada interna acima descrito.

A Figura 4.5 mostra detalhadamente como as soluções retornadas pelo AG são selecionadas, durante o procedimento de validação cruzada interna.

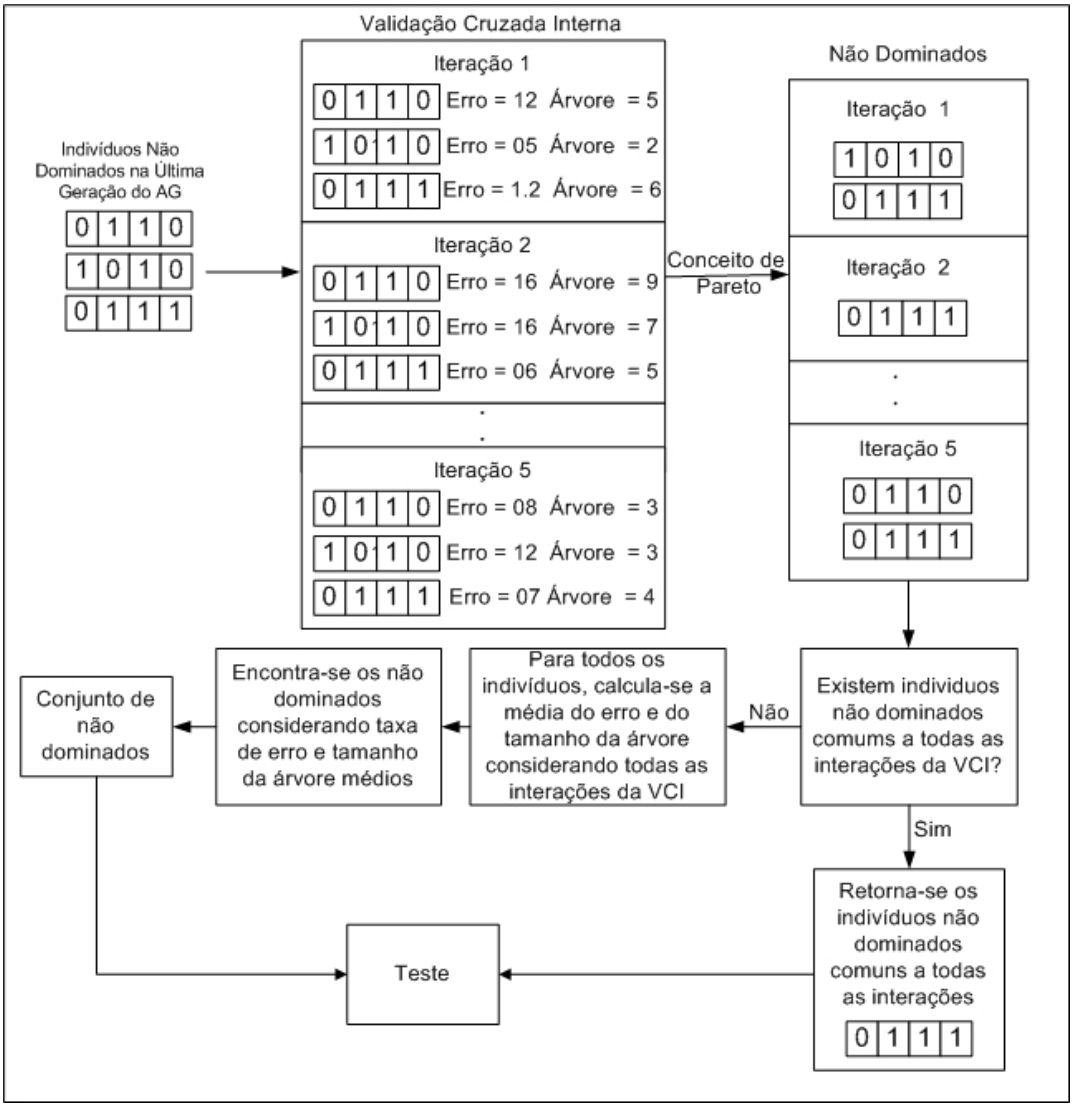


Figura 4.5: Escolha das soluções do AG através da Validação Cruzada Interna

Conforme mencionado anteriormente, na última geração, encontrados os indivíduos não dominados, estes são submetidos a um processo de validação cruzada interna. Na Figura 4.5 são três esses indivíduos. Para cada iteração da validação cruzada interna, que na Figura 4.5 possui fator 5, os indivíduos são avaliados (através do erro e tamanho da árvore gerados pelo C4.5) e depois comparados em relação ao conceito de dominância de Pareto. Aqueles indivíduos que permanecerem não dominados após essa comparação são inseridos em uma lista de não dominados.

Assim, seguindo o exemplo da Figura 4.5, terminadas as 5 interações da validação cruzada, temos 5 conjuntos de indivíduos não dominados. Através desses conjuntos é preciso determinar quais indivíduos serão retornados como soluções para o AG.

O primeiro critério de escolha é extremamente qualitativo. São retornados os indivíduos que permanecerem não dominados em todas as partições da validação cruzada interna, ou seja, os indivíduos que estiverem presentes em todos os conjuntos de não dominados. Assim, no exemplo da Figura 4.5, apesar das interações 3 e 4 não serem explicitamente relatadas no retângulo que indica os conjuntos de indivíduos não dominados retornados, consideramos que o indivíduo 0111 apareceu em todas as iterações, e por isso foi retornado como solução para o problema.

Porém, em alguns casos não encontramos nenhum indivíduo comum aos conjuntos de não dominados de todas as partições (Ou seja, considerando os conjuntos de não dominados, retornados por cada iteração da validação cruzada interna, representados na Figura 4.5 por um retângulo nomeado não-dominados, não existe nenhum indivíduo comum a todos os conjuntos). Nesse caso, um critério estatístico é utilizado para retornar um conjunto de soluções. Uma média do tamanho da árvore e da taxa de erro de todos os indivíduos, considerando todas as partições, é calculada, e essa taxa de erro média e o tamanho da árvore médio de cada indivíduo são então utilizados para encontrar um conjunto de indivíduos não dominados.

Os indivíduos retornados, utilizando tanto o critério qualitativo quanto o critério estatístico, são então submetidos ao teste, no qual a partição reservada para esse propósito será utilizada para estimar a taxa de erro e tamanho da árvore dos indivíduos retornados em dados ainda não vistos.

Como já citado, além do procedimento de validação cruzada interna, realizado para escolher os indivíduos que serão retornados pelo AGMO, um procedimento de validação cruzada é também utilizado para estimar a qualidade de cada indivíduo do AG. Ou seja, o AGMO é executado p vezes, onde p é o número de partições que serão utilizadas durante a validação cruzada.

Em cada uma das p vezes em que o AGMO é executado, um procedimento de validação cruzada interna deve também ser realizado, para escolher os indivíduos retornados. Porém, às p execuções do AGMO para uma única base de dados, apenas o critério qualitativo ou o critério estatístico pode ser aplicado. Isso acontece porque,

finalizadas as *p* iterações do AG, uma média de seus resultados é retornada, e essa média deve ser baseada em números gerados utilizando um mesmo critério, como mostra a Figura 4.6.

Dessa forma, se para uma dada base de dados o critério qualitativo é utilizado em menos de 50% das partições da validação cruzada "externa", as soluções são retornadas de acordo com o critério estatístico.

A Figura 4.6 mostra as execuções do AG em um procedimento de validação cruzada com 5 partições. Após o AG ser executado, um conjunto de soluções é retornado para cada partição. Dessas soluções, alguns dados estatísticos são extraídos, como o número total de indivíduos retornados pelo AGMO (representado na figura como TotalE ou TotalQ, onde E representa uma solução retornada através do critério estatístico e Q através do critério qualitativo, como demonstrado na Figura 3.5), o número de indivíduos que dominam ou são dominados pela solução padrão e indivíduos neutros. Finalizado o processo de validação cruzada, uma média desses números deve ser calculada, para avaliar a qualidade das soluções produzidas em todas as iterações da validação cruzada.

A Figura 4.6 (a) descreve uma base de dados em que o critério qualitativo foi utilizado em 4 das 5 partições da validação cruzada interna. Nesse caso, como 4 corresponde a mais de 50% das 5 partições disponíveis, a média dos valores estatísticos estimados pelo algoritmo, é calculada utilizando apenas as partições em que o critério qualitativo foi utilizado. Assim, a média do total de indivíduos retornados para a Figura 4.6 (a) seria calculada pela Fórmula 4.1.

$$Total_{Media} = \frac{TotalQ2 + TotalQ3 + TotalQ4 + TotalQ5}{4} \qquad (4.1)$$

Já a Figura 4.6(b) representa uma base em que o critério qualitativo é utilizado em apenas duas partições. Nesse caso, o cálculo da média dos dados estatísticos seria representada pela Fórmula 4.2.

$$Total_{Media} = \frac{\sum_{i=1}^{6} TotalE(i)}{5} \qquad (4.2)$$

Observe que, na Fórmula 4.2, utilizamos o total retornado utilizando o critério estatístico para todas as bases. Os dados qualitativos são ignorados.

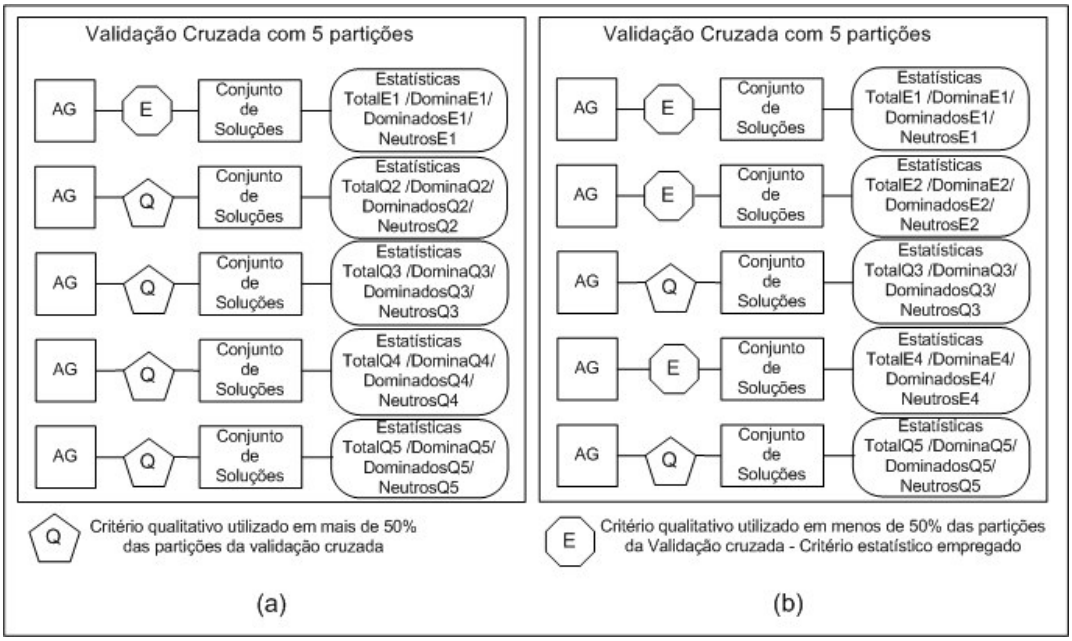


Figura 4.6: Método utilizado pra retornar o valor final da média da taxa de erro e tamanho da árvore gerados pelo AG em todas as partições da validação cruzada

4.1.5 Implementação

O Algoritmo 4.1 mostra o pseudocódigo do método implementado:

```
Cria população inicial
PARA CADA geração FAÇA
 PARA CADA Indivíduo FAÇA
  Executa o C4.5 com o subconjunto de atributos representado
  pelo indivíduo
  Armazena o resultado dos objetivos utilizados pela função de
  avaliação // taxa de erro e tamanho da árvore
 FIM PARA
 Encontra todos os indivíduos não dominados da população
 Adiciona Nelit indivíduos não dominados na população da próxima
 geração //N <= (tamanho da população/2)
 PARA i ← 1 to (N - Nelit)/2 FAÇA
  Executa a seleção por torneio duas vezes, para selecionar dois
  indivíduos pais, P1 e P2
  Executa o cruzamento de P1 e P2, produzindo filhos F1 e F2
  Realiza mutação em F1 e F2
  Adiciona F1 e F2 na população da próxima geração
 FIM PARA
FIM PARA
Avalia os indivíduos da última geração
Encontra todos os indivíduos não dominados da última geração
Realiza um procedimento de Validação Cruzada(VC)
Retorna apenas os indivíduos não dominados em todas as iterações
da VC
```

Algoritmo 4.1. Pseudocódigo do AGMO proposto

O tamanho da população utilizada pelo AG foi de 100 indivíduos, que evoluem durante 50 gerações.

Os resultados obtidos pelo método proposto podem ser encontrados no Capítulo 5.

4.2 Seleção Seqüencial para frente Multiobjetiva (SSFMO)

O algoritmo de seleção seqüencial para frente (SSF) vem sendo utilizado há muitos anos, na tentativa de resolver a tarefa de seleção de atributos, como mostrado em [Caruana e Freitag 1994], [Koller e Sahami 1996]. O algoritmo inicia sua execução com um conjunto vazio de atributos, e a cada iteração adiciona o melhor atributo a esse conjunto, até que ele não possa mais ser melhorado. Atributos inseridos na iteração i não podem ser posteriormente removidos nas iterações i+1, i+2,...

Como qualquer método de seleção de atributos, utilizar a seleção seqüencial para frente traz vantagens e desvantagens. Entre as principais vantagens estão sua facilidade de implementação e seu baixo custo computacional. Entre as principais desvantagens, podemos citar a sua ineficácia em lidar com a interação entre os atributos, que não permite que esses métodos sejam capazes de resolver problemas como o citado na Seção 2.2.

Com o objetivo de suprir as deficiências acima citadas, sem deixar de lado as vantagens que o SSF traz, muitos estudos trazem variações desse algoritmo, como mostrado em [Aha e Bankert 1996]. Esses estudos comprovam que pequenas alterações no algoritmo básico levam a um grande aumento da previsão preditiva do classificador.

Pensando nisso, propomos nesse trabalho uma versão multiobjetiva do algoritmo de SSF, o SSFMO.

Sabemos que qualquer algoritmo multiobjetivo é caracterizado principalmente por, ao invés de retornar apenas uma solução ótima, retornar um conjunto de soluções ótimas. Assim, o primeiro desafio de transformar um SSF simples em um SSFMO é fazer com que ele retorne múltiplas soluções.

Para isso, foi incorporada ao algoritmo uma lista de soluções não dominadas, que deve guardar, de todas as soluções geradas a cada iteração do algoritmo, as que são não dominadas de acordo com o conceito de Pareto. Essa idéia é inspirada em vários AGMOs presentes na literatura que utilizam como artifício uma população externa de soluções, que armazena todas as soluções não dominadas até o momento, como o SPEA

[Zitzler e Thiele 1999], que guarda nessa lista todas as soluções geradas, desde a primeira até a última geração.

O Algoritmo 4.2 apresenta o pseudocódigo do método proposto:

```
Cria lista de soluções não dominadas, inicializada com o
conjunto vazio
Gera as soluções iniciais utilizando apenas um atributo
ENQUANTO Lista de soluções estiver sendo atualizada
 PARA CADA solução FAÇA
  Execute o C4.5 com o subconjunto de atributos representado
  pelo indivíduo
  Armazene o resultado dos objetivos utilizados pela função de
  avaliação // taxa de erro e tamanho da árvore
 FIM PARA
 Dentre as soluções criadas, encontra as soluções não
 dominadas
 Realiza um procedimento de validação cruzada (VC)
 Atualiza a lista de soluções // insere novas soluções e
 //retira as antigas soluções dominadas pelas novas
 Gera novas soluções a partir das soluções presentes na lista
 de não dominadas
FIM ENQUANTO
```

Algoritmo 4.2**.** Pseudocódigo do SSFMO proposto

Como mostra o Algoritmo 4.2, a versão multiobjetiva do algoritmo é iniciada da mesma forma que a versão tradicional: um conjunto de soluções iniciais é gerado e avaliado. As diferenças começam a aparecer na hora da avaliação. Como no AGMO descrito na Seção 4.1, dois objetivos devem ser otimizados durante a busca: a taxa de erro e o tamanho da árvore gerados pelo C4.5 durante o treinamento. Por isso, o valor desses dois critérios é armazenado e posteriormente utilizado no julgamento de uma solução como sendo melhor ou pior que outra.

Após a avaliação das soluções, com base nos conceitos de dominância de Pareto, são selecionadas as soluções não dominadas. Essas soluções são então utilizadas para atualizar a lista de soluções não dominadas. Essa atualização consiste em comparar as soluções da lista com as não dominadas da iteração atual de acordo com os conceitos de Pareto, e deixar que permaneçam na lista apenas as soluções não dominadas.

É a partir dessa lista atualizada que são geradas as novas soluções. A cada uma das soluções da lista um novo atributo, ainda não presente, é inserido a ela.

Considere, por exemplo, um conjunto de atributos $\{A_1, A_2, A_3, A_4\}$. Inicialmente cada um desses atributos gera uma solução. Suponhamos que das 4 soluções iniciais, baseados nas taxas de erro e tamanho da árvore geradas pelo C4.5, apenas a solução S_2, correspondente ao atributo A_2, seja considerada não dominada. Após inserida na lista de

soluções não dominadas, ela é então combinada com todos os atributos disponíveis. Assim, 3 novas soluções são geradas: $S_{21} = \{A_2, A_1\}$, $S_{23} = \{A_2, A_3\}$e $S_{24} = \{A_2, A_4\}$. Essas soluções são então novamente avaliadas, e S_{21} e S_{24} consideradas não dominadas. Elas são então comparadas a S_2, que já está na lista. Se elas não dominarem e nem forem dominadas por S_2 , a lista de soluções não dominadas passa a ter 3 soluções: S_2, S_{21} e S_{24} . Como na iteração anterior, novas soluções são geradas tendo por base S_{21} e S_{24}, gerando $S_{213} = \{A_2, A_1, A_3\}$, $S_{214} = \{A_2, A_1, A_4\}$, e $S_{243} = \{A_2, A_4, A_3\}$. Esses subconjuntos são novamente avaliados, e apenas a solução S_{214} inserida no conjunto dos não dominados. Na hora de atualizar a lista de não dominados, suponhamos que os valores de erro e tamanho da árvore de S_{214} são dominados pelos valores de S_{24}. Nesse caso a lista não é atualizada, e são retornadas para o usuário como soluções S_2, S_{21} e S_{24}.

Da mesma forma que para o AGMO, não temos usuários no sentido nobre do termo. Assim, os resultados são apresentados ao usuário da mesma forma que são apresentados pelo AGMO: contabilizando, dentre as soluções retornadas pela última geração, aquelas que dominam a solução padrão do problema (em relação ao erro e ao tamanho da árvore gerados pelo C4.5), aquelas que são dominadas pela solução padrão, e as que não apresentam relação de dominância. Entende-se por solução padrão o conjunto de todos os atributos originais.

5. EXPERIMENTOS E RESULTADOS

Os experimentos foram realizados utilizando-se um procedimento de validação cruzada, como descrito na Seção 4.1.4.

As bases de dados usadas nos experimentos foram divididas aleatoriamente em 10 partições, conservando a mesma distribuição de classes para cada uma delas (validação cruzada estratificada). 9/10 dos dados foram destinados ao treinamento e 1/10 para o teste. Porém, a utilização da abordagem *wrapper* exigiu que os 9/10 destinados ao treinamento fossem novamente separados, sendo destinados 8/9 para construção da árvore e 1/9 para avaliação da árvore construída pelo C4.5.

Os experimentos foram realizados utilizando 18 bases de dados públicas disponíveis na UCI (University of California at Irvine) [Murphy e Aha 1994]. O número de exemplos, atributos e classes para cada base de dados estão relacionados na Tabela 5.1.

Os algoritmos descritos neste capítulo foram implementados em Java e utilizaram a versão 8.0 do C4.5. Os experimentos foram realizados em um servidor com dois processadores de 1.1 GHz e 3 Gb de memória.

Dois tipos de experimentos envolvendo métodos de seleção de atributos foram realizados: os primeiros considerando a otimização simples e segundo a otimização multiobjetiva. A Seção 5.1 descreve os experimentos de otimização simples, realizados utilizando o C4.5, o algoritmo de Seleção Seqüencial para frente (SSF) e um algoritmo genético que avalia os mesmos objetivos do AGMO proposto, considerando-os em uma função ponderada.

A Seção 5.2 descreve experimentos realizados utilizando o algoritmo genético multiobjetivo proposto, a versão do SSF multiobjetiva também proposta e o SPEA (*Strength Pareto Evolutionary Algorithm*), previamente descrito na Seção 3.7.4.

Os resultados iniciais desses experimentos foram publicados em [Pappa *et* al 2002a] e [Pappa *et* al 2002b].

Tabela 5.1: Características das bases de dados utilizadas nos experimentos

Base de Dados	Número de Atributos	Número de Registros	Número de Classes
Arrhythmia	269	452	16
Balance-Scale	4	625	3
Bupa	6	345	2
Car	6	1717	4
Crx	15	690	2
Dermatology	34	366	6
Glass	10	214	7
Ionosphere	34	351	2
Iris	4	150	3
Mushroom	22	8124	2
Pima	8	768	2
Promoters	57	106	2
Sick-euthyroid	25	3163	2
Tic tac toe	9	958	2
Vehicle	18	846	4
Votes	16	435	2
Wine	13	178	3
Wiscosin breast-cancer	9	699	2

5.1 Experimentos envolvendo métodos de Otimização Simples

Esse trabalho propõe a avaliação do comportamento de métodos multiobjetivos de seleção de atributos, através de um algoritmo genético e uma nova versão do tradicional método de seleção seqüencial para frente (SSF). Antes de se apresentar os experimentos para a versão multiobjetiva desses métodos, suas versões utilizando otimização simples foram implementadas e avaliadas.

Tanto a versão simples do AG quanto do SSF utilizam uma função de avaliação que combina os dois objetivos a serem alcançados, através da fórmula:

$$\textit{função de avaliação} = \frac{3}{4}\textit{erro} + \frac{1}{4}\textit{tamanho da árvore} \qquad (5.1)$$

Os pesos 3 e 1, atribuídos respectivamente ao erro e tamanho da árvore na Fórmula 5.1, são valores padrão utilizados na literatura de seleção de atributos e algoritmos genéticos. Costuma-se atribuir um peso maior ao erro, apesar de sabermos que o tamanho da árvore gerada é também de extrema importância.

A versão simples do AG utiliza os mesmos operadores e a mesma estratégia de seleção (porém sem o conceito de dominância de Pareto) da versão multiobjetiva, sendo o elitismo utilizado apenas para conservar o melhor indivíduo da população.

Além disso, o C4.5, um dos algoritmos de aprendizagem mais utilizados na literatura, em se tratando da tarefa de seleção de atributos utilizando a abordagem *wrapper*, foi utilizado como método base para as comparações que serão feitas nesse capítulo.

As Tabelas 5.2 e 5.3 mostram os resultados dos experimentos utilizando as 18 bases descritas na Tabela 5.1. A Tabela 5.2 relata a média das taxas de erro e a Tabela 5.3 a média do tamanho das árvores geradas pelo C4.5, pelo SSF Simples (descrito na Seção 2.2) e pelo AG Simples nas 10 interações da validação cruzada. Em todas as tabelas apresentadas nesse capítulo, os números após o sinal "±" representam os desvios padrão obtidos.

Sinais de (+) ou (-) após os desvios padrão de qualquer campo de qualquer tabela deste capítulo indicam que os resultados obtidos são estatisticamente melhores (+) ou piores (-) que os obtidos pelo C4.5. Campos sem nenhuma indicação representam situações em que não se pode concluir, em termos estatísticos, que um método é melhor do que o outro.

Um resultado *i* é estatisticamente melhor que um resultado *j*, se a (média + desvio padrão) de *i* tem valor inferior a (média - desvio padrão) de *j*, sendo a média de *i* menor que a média de *j.* Consideremos a base *balance- scale*, por exemplo. A taxa de erro do C4.5 é de 36.34 ± 1.08, enquanto para o SSF ela é de 33.12 ± 1.41. A menor média, que é a do SSF, somada a seu desvio padrão tem valor 34.56. Já a média do C4.5 menos o seu desvio padrão tem valor 35.26. Como 33.12 é menor que 34.56, podemos dizer que a taxa de erro obtida pelo SSF é estatisticamente melhor que a do C4.5.

Porém, no caso da base *arrhythmia*, por exemplo, a taxa de erro do C4.5 é de 32.93 ± 3.11 e a do SSF de 37.37 ± 1.42. Nesse caso, 36.04 (32.93 + 3.11) é maior que 35.95 (37.37 – 1.42), e os resultados são estatisticamente equivalentes. Assim, não podemos dizer que o resultado do C4.5 é estatisticamente melhor que o SSF, e sim que ocorre um "empate estatístico".

Analisando a Tabela 5.2, observamos que o SSF perde para o C4.5 em 8 bases, em relação ao erro de classificação. Dessas, em *car*, *promoters* e *tic-tac-toe*, o erro obtido pelo SSF é no mínimo o dobro que o obtido pelo C4.5. É interessante observar, como relatado na Tabela 5.4, que nessas 3 bases o número de atributos selecionados é em média 1. Uma possível explicação para isso é o fato dessas 3 bases apresentarem forte interação entre alguns atributos, o que não pode ser detectado por algoritmos de

busca seqüenciais. Por uma pequena diferença o SSF é melhor que o C4.5 em *balance-scale*. Nas outras 9 bases, nenhum dos algoritmos pode ser considerado estatisticamente melhor do que o outro.

Já o AG simples perde para o C4.5 em 7 bases. Essas bases são as mesmas em que o C4.5 ganha do SSF. Porém, as diferenças entre as taxas de erros, na maioria dos casos, não são tão grandes quanto as obtidas comparando o C4.5 com o SSF. O AG também só obteve taxa de erro menor que a do C4.5 em *balance-scale*.

Tabela 5.2: Taxas de erro obtidas utilizando o C4.5, o SSF Simples e o AG Simples nos experimentos realizados

Base de Dados	Taxa de Erro		
	C4.5	SSF Simples	AG Simples
Arrhythmia	32.93 ± 3.11	37.37 ± 1.42	30.83 ± 2.1
Balance-Scale	36.34 ± 1.08	33.12 ± 1.41 (+)	33.62 ± 1.56 (+)
Bupa	37.07 ± 2.99	41.08 ± 2.86	37.68 ± 1.73
Car	7.49 ± 0.70	29.58 ± 0.15 (-)	15.12 ± 1.1 (-)
Crx	15.95 ± 1.43	14.46 ± 1.48	13.33 ± 1.34
Dermatology	6.0 ± 0.98	9.61 ± 2.14 (-)	6.82 ± 1.45
Glass	1.86 ± 0.76	2.36 ± 0.78	2.36 ± 0.79
Ionosphere	10.2 ± 1.25	12.46 ± 1.6	10.76 ± 1.66
Iris	6.0 ± 2.32	6.01 ± 2.09	7.34 ± 2.52
Mushroom	0.0 ± 0.0	0.28 ± 0.12 (-)	0.01 ± 0.01 (-)
Pima	26.07 ± 1.03	31.03 ± 0.31 (-)	30.5 ± 1.72 (-)
Promoters	16.83 ± 2.55	32.84 ± 5.88 (-)	25.67 ± 4.27 (-)
Sick-euthyroid	2.02 ± 0.12	2.38 ± 0.29	2.38 ± 0.26
Tic tac toe	15.75 ± 1.4	31.61 ± 1.77 (-)	24.82 ± 1.83 (-)
Vehicle	26.03 ± 1.78	34.93 ± 2.11 (-)	31.1 ± 1.18 (-)
Votes	3.2 ± 0.91	4.35 ± 0.92	4.57 ± 0.89
Wine	6.69 ± 1.82	8.87 ± 1.88	6.66 ± 1.59
Wiscosin breast cancer	5.28 ± 0.95	7.88 ± 1.02 (-)	7.59 ± 0.91 (-)
Sumário do comportamento dos algoritmos em relação ao C4.5			
Número de bases em que houve melhora		1	1
Número de bases em que houve piora		8	7

Em relação ao tamanho da árvore, relatado na Tabela 5.3, todas as árvores construídas pelo SSF têm tamanho menor que as do C4.5, com exceção da árvore construída para a base de dados *glass*. Já o AG ganha do C4.5 em 14 bases e empata nas outras quatro.

A Tabela 5.4 mostra o número de atributos selecionados pelo SSF Simples e pelo AG Simples.

Tabela 5.3: Tamanho das árvores obtidas utilizando o C4.5, o SSF Simples e o AG Simples nos experimentos realizados

Base de Dados	Tamanho da árvore		
	C4.5	SSF Simples	AG Simples
Arrhythmia	80.2 ± 2.1	31.4 ± 5.35 (+)	78.6 ± 2.89
Balance-Scale	41.0 ± 1.29	12.5 ± 2.36 (+)	14.0 ± 2.38 (+)
Bupa	44.2 ± 3.75	10.8 ± 3.90 (+)	14.8 ± 4.79 (+)
Car	165.3 ± 2.79	1.0 ± 0 (+)	52.4 ± 5.93 (+)
Crx	29.0 ± 3.65	3.0 ± 0 (+)	15.3 ± 1.99 (+)
Dermatology	34.0 ± 1.89	21.6 ± 0.52 (+)	24.2 ± 1.04 (+)
Glass	11.0 ± 0.0	11.0 ± 0	11.0 ± 0
Ionosphere	26.2 ± 1.74	10.2 ± 1.49 (+)	16.8 ± 2.09 (+)
Iris	8.2 ± 0.44	5.0 ± 0 (+)	6.0 ± 0.68 (+)
Mushroom	32.7 ± 0.67	24.5 ± 1.53 (+)	28 ± 1.23 (+)
Pima	45.0 ± 2.89	5.0 ± 1.40 (+)	13.6 ± 4.86 (+)
Promoters	23.8 ± 1.04	5 ± 0 (+)	13.4 ± 2.42 (+)
Sick-euthyroid	24.8 ± 0.69	7.4 ± 0.65 (+)	14.6 ± 2.06 (+)
Tic tac toe	130.3 ± 4.25	4.6 ± 0.75 (+)	49.9 ± 9.57 (+)
Vehicle	134.0 ± 6.17	88.8 ± 11.02 (+)	130.8 ± 6.67
Votes	10.6 ± 0.26	3.0 ± 0 (+)	5.6 ± 1.08 (+)
Wine	10.2 ± 0.68	9.0 ± 0.3 (+)	11.0 ± 1.84
Wiscosin breast cancer	28.0 ± 2.13	16.0 ± 1.66 (+)	23 ± 1.33 (+)
Sumário do comportamento dos algoritmos em relação ao C4.5			
Número de bases em que houve melhora		17	14
Número de bases em que houve piora		0	0

Tabela 5.4: Número de atributos selecionados nos experimentos pelo SSF Simples e pelo AG Simples

Base de Dados	# Atributos Selecionados	
	SSF Simples	AG Simples
Arrhythmia	5.2 ± 0.44	128 ± 11.2
Balance-Scale	1.5 ± 0.16	1.5 ± 0.16
Bupa	2.1 ± 0.31	2.8 ± 0.36
Car	1 ± 0	4.2 ± 0.2
Crx	1 ± 0	6.7 ± 0.47
Dermatology	6.1 ± 0.31	16.8 ± 0.84
Glass	1.2 ± 0.13	5.1 ± 0.35
Ionosphere	2.7 ± 0.3	14.4 ± 0.6
Iris	1 ± 0	2.3 ± 0.21
Mushroom	3.6 ± 0.43	11.6 ± 0.93
Pima	1.6 ± 0.34	3.5 ± 0.34
Promoters	1 ± 0	26 ± 0.93
Sick-euthyroid	3.1 + 0.35	9.9 ± 0.79
Tic tac toe	1.2 ± 0.22	5.0 ± 0.42
Vehicle	4.2 ± 0.5	9.0 ± 0.54
Votes	1 ± 0	7.1 ± 0.66
Wine	3.0 ± 0.3	5.6 ± 0.62
Wiscosin	1.5 ± 0.16	3.4 ± 0.5

Em relação à Tabela 5.4, comparando o número de atributos selecionados pelo AG Simples e pelo SSF, percebemos que a média de atributos selecionados pelo SSF é

menor que a dos selecionados pelo AG simples para a maioria das bases. Mas nem sempre esse conjunto menor de atributos oferece taxas de erros também menores que as obtidas pelo C4.5 utilizando todos os atributos.

De modo geral, tanto o AG simples quanto o SSF simples tiveram um desempenho surpreendentemente ruim. Nos experimentos realizados neste trabalho, esses métodos pioraram a taxa de erro com mais freqüência do que a melhoraram, em comparação com a solução padrão, consistindo de todos os atributos originais.

5.2 Experimentos envolvendo métodos de Otimização Multiobjetiva

A Seção 5.1 mostrou comparações entre os resultados obtidos pelo SSF e pelo AG simples em relação ao C4.5. Esta seção mostra como as versões multiobjetivas desses dois métodos se comportam em relação ao C4.5.

Como já colocado, uma das características que diferem um método de otimização simples de um método multiobjetivo é o número de soluções retornadas por ele. Enquanto a otimização simples tem como resposta apenas uma solução, a otimização multiobjetivo oferece ao usuário múltiplas soluções, para que este escolha a que mais se adapta ao problema que ele pretende resolver. Também como já colocado, o sistema aqui proposto não possui um usuário no sentido nobre do termo. Por isso, o conjunto de soluções não dominadas retornado tanto pelo AGMO quanto pelo SSFMO precisa ser sumarizado para que possa ser diretamente comparado com a solução que representa os atributos originais da base de dados. Cabe ressaltar que avaliar a qualidade das soluções retornadas por um algoritmo de otimização multiobjetivo é bastante difícil [Zitzler *et* al 2002], [Knowles e Corne 2002].

A primeira idéia foi utilizar a média das soluções não dominadas como forma de sumarização, mas esse tipo de medida vai totalmente contra o princípio de otimização multiobjetiva, pois transforma múltiplas soluções em uma solução média.

Para resolver esse problema, durante a sumarização, conforme mencionado anteriormente, são contabilizadas, dentre as soluções retornadas pela última geração, aquelas que dominam a solução padrão do problema (em relação ao erro e ao tamanho da árvore gerados pelo C4.5), aquelas que são dominadas pela solução padrão e as que não apresentam relação de dominância. Entende-se por solução padrão o conjunto de todos os atributos originais.

Esses números refletem se o AGMO ou o SSFMO foram capazes de encontrar soluções com taxas de erro e tamanho da árvore menores do que os obtidos pela solução padrão.

A Tabela 5.5 sumariza os resultados encontrados pelo AGMO. A segunda coluna mostra o número total de soluções encontradas. A terceira, quinta e sétima colunas mostram o número de soluções do AG que dominam a solução padrão do C4.5, o número de soluções que são dominadas por ela e o número de indivíduos neutros, que não dominam nem são dominados pela solução padrão. As colunas que seguem as descritas anteriormente apresentam as freqüências relativas dos números apresentados. Por exemplo, na base *Arrhythmia* o valor de F_{Domina} é 0.21, o qual foi obtido dividindo-se 0.8 (valor da coluna *Domina*) por 3.9 (valor da coluna *Total*).

Tabela 5.5: Resultados dos experimentos utilizando o AGMO

	Soluções geradas pelo AG						
Base de Dados	Total	Domina	F_{Domina}	Dominado	$F_{Dominado}$	Neutros	$F_{Neutros}$
Arrhythmia	3.9 ± 0.54	0.8 ± 0.38	0.21	1.3 ± 0.68	**0.33**	1.8 ± 0.44	0.46
Balance-Scale	1.0 ± 0.0	0.7 ± 0.15	**0.7**	0	0	0.3 ± 0.15	0.3
Bupa	6.1 ± 0.38	1.9 ± 0.83	0.31	0	0	4.2 ± 1.03	0.69
Car	38.3 ± 0.76	0.1 ± 0.1	0.002	0	0	38.2 ± 0.77	**0.998**
Crx	4.55 ± 0.67	2.55 ± 0.69	**0.56**	0.22 ± 0.15	0.05	1.77± 0.77	0.39
Dermatology	1.11 ± 0.11	0.88 ± 0.17	**0.8**	0	0	0.22 ± 0.11	0.2
Glass	46.9 ± 1.03	0	0	2.8 ± 2.70	**0.06**	44.1 ± 2.56	**0.94**
Ionosphere	1.14 ± 0.14	0.42 ± 0.2	0.37	0.14 ± 0.14	0.12	0.57 ± 0.3	0.5
Iris	4.4 ± 0.16	3.5 ± 0.52	**0.8**	0.1 ± 0.1	0.02	0.8 ± 0.46	0.18
Mushroom	1.9 ± 0.18	1.3 ± 0.15	**0.68**	0	0	0.6 ± 0.22	0.32
Pima	18.3 ± 1.15	6.2 ± 1.77	0.34	0.1 ± 0.1	0	12.0 ± 1.42	0.66
Promoters	1.5 ± 0.16	0.5 ± 0.22	0.33	0	0	1.0 ±0.26	0.67
Sick- euthyroid	25.4 ± 0.93	0.4 ± 0.22	0.02	0.6 ± 0.22	0.02	24.4 ± 0.85	**0.96**
Tic tac toe	16.5 ± 1.0	0.1 ± 0.1	0	0	0	16.4 ± 1.01	**1**
Vehicle	6.1 ± 0.76	1.5 ± 0.43	0.25	1.1 ± 0.46	0.18	3.5 ± 0.82	0.57
Votes	26.6 ± 1.63	16.0 ± 4.57	**0.6**	0	0	10.6 ± 4.1	0.4
Wine	4.66 ± 1.21	2.22 ± 1.22	0.48	1.44 ± 0.60	0.31	1.0 ± 0.41	0.21
Wiscosin	9.3 ± 0.4	4.6 ± 1.33	0.5	1.9 ± 0.91	0.2	2.8 ± 0.89	0.3

Uma forma de avaliar os resultados da Tabela 5.5 é analisar os valores da coluna F_{domina}, contendo a freqüência relativa do número de soluções encontradas pelo AGMO consideradas melhores que a solução padrão, considerando tanto a taxa de erro quanto o tamanho da árvore. Em seis bases - *balance-scale*, *crx*, *dermatology*, *iris*, *mushroom* e *votes* – o valor de F_{domina} é maior ou igual a 0.5, ou seja, 50% ou mais das soluções encontradas pelo AG dominam a solução padrão constituída por todos os atributos.

Apenas nas bases *glass* e *tic-tac-toe* nenhuma solução encontrada pelo AG dominou a solução padrão. Porém, em *tic-tac-toe*, a solução padrão também não dominou nenhuma solução encontrada pelo AG, sendo ela a única base em que todas as soluções encontradas eram neutras.

Em *car* o número de soluções neutras também foi praticamente 100%, pois apenas 0.2% das soluções encontradas eram melhores que a padrão. Um número alto de soluções neutras pode ser também observado em *glass* e *sick-euthyroid.*

Em metade das bases utilizadas nos experimentos, nenhuma das soluções encontradas foi dominada pela solução padrão, ou seja, nenhuma das soluções encontradas era pior que a solução padrão com todos os atributos.

Arrhythmia e *glass* são as únicas bases em que o valor de $F_{dominado}$ é maior do que o valor de F_{domina}, indicando que nessas duas bases o AG não foi bem sucedido. Porém, em *glass* o valor de $F_{dominado}$ é pequeno, indicando 6% das soluções encontradas como sendo piores que a solução padrão contra 94% de soluções neutras. Já em *arrhythmia* o número de soluções encontradas piores que a solução padrão sobe para 33%, contra 21% de soluções melhores que a padrão e 46% neutras.

Sumarizando, temos 14 bases em que as soluções encontradas pelo AG dominam a solução padrão com mais freqüência (ou seja, o valor da coluna Domina é maior que o valor da coluna Dominado) do que vice-versa. Considerando então as quatro bases restantes dentre as 18 analisadas (*tic-tac-toe, glass, sick-euthyroid* e *arrhythmia*), no *tic-tac-toe, sick-euthyroid* e em *glass* praticamente todas as soluções encontradas pelo AG são soluções neutras. Apenas em *Arrhythmia* o número de soluções encontradas pelo AG é, em sua maioria, dominada pela solução padrão, mas ainda assim esse número é inferior ao total de soluções neutras.

Comparando então o AGMO proposto com o AG simples, que encontrou taxas de erro maiores que a solução padrão com todos atributos em 7 bases, podemos concluir que a utilização de algoritmos genéticos multiobjetivos, no problema de seleção de atributos, tem um desempenho muito melhor que algoritmos genéticos utilizando técnicas de otimização simples.

A Tabela 5.6 mostra o resultado dos experimentos com o SSFMO. Nesse caso, são 8 as bases cujo número de soluções encontradas pelo AG dominando a solução padrão ultrapassa 50% (*arrhythmia*, *balance-scale*, *bupa*, *crx*, *glass*, *iris*, *pima* e *wine*). Note que apenas em duas bases o número de soluções encontradas pelo SSFMO

dominadas pela solução padrão foi diferente de 0: em *wine* e *wisconsin*, sendo que no primeiro caso o número de soluções encontradas pelo SSFMO dominadas pela solução padrão equivale a apenas 1% do total de soluções encontradas. Foram duas também as bases em que apenas soluções neutras foram encontradas: *dermatology* e *mushroom*.

Tabela 5.6: Resultados dos experimentos utilizando o SSFMO

	Soluções geradas pelo SSFMO						
Base de Dados	Total	Domina	F_{Domina}	Dominado	$F_{Dominado}$	Neutros	$F_{Neutros}$
Arrhythmia	32.2 ± 10.82	17.4 ± 9.38	**0.54**	0	0	14.8 ± 8.68	0.46
Balance-Scale	1.8 ± 0.2	0.9 ± 0.23	**0.5**	0	0	0.9 ± 0.1	0.5
Bupa	2.9 ± 0.31	1.9 ± 0.48	**0.65**	0	0	1.0 ± 0.45	0.35
Car	4.3 ± 0.33	0.3 ± 0.15	0.07	0	0	4.0 ± 0.33	0.93
Crx	84.1 ± 2.05	75.0 ± 8.55	**0.89**	0	0	9.1 ± 9.1	0.11
Dermatology	76.5 ± 10.3	0	0	0	0	76.5 ± 10.3	1
Glass	94.1 ± 5.24	93.1 ± 5.24	**0.99**	0	0	1.0 ± 0	0.01
Ionosphere	12.9 ± 6.23	1.8 ± 1.21	0.14	0	0	11.1 ± 5.21	0.86
Iris	3.5 ± 0.34	3.0 ± 0.54	**0.86**	0	0	0.5 ± 0.27	0.14
Mushroom	51.9 ± 11.88	0	0	0	0	51.9 ± 11.88	1
Pima	11.1 ± 1.88	10.6 ± 1.97	**0.95**	0	0	0.5 ± 0.27	0.05
Promoters	66.6 ± 12.66	18.2 ± 11.43	0.27	0	0	48.4 ± 14.16	0.73
Sick- euthyroid	50.3 ± 6.44	4.9 ± 1.89	0.1	0	0	45.4 ± 6.74	0.9
Tic tac toe	8.1 ± 1.54	0.9 ± 0.28	0.11	0	0	7.2 ± 1.41	0.89
Vehicle	3.6 ± 0.16	0.6 ± 0.16	0.17	0	0	3.0 ± 0.15	0.83
Votes	98.4 ± 0.37	9.9 ± 9.68	0.1	0	0	88.5 ± 9.84	0.9
Wine	8.3 ± 6.1	7.6 ± 5.97	**0.92**	0.1 ± 0.1	0.01	0.6 ± 0.22	0.07
Wiscosin	10.1 ± 4.76	4.6 ± 2.84	0.45	3.7 ± 3.18	0.37	1.8 ± 0.63	0.18

Assim, o desempenho do SSFMO também se mostra muito superior ao SSF tradicional (otimização com único objetivo). Nas três bases em que o SSF tradicional teve seu pior desempenho (*car*, *promoters* e *tic-tac-toe*), o SSFMO consegue encontrar soluções melhores que a padrão, apesar do número de soluções neutras ainda predominar.

Apenas para conveniência do leitor, a Tabela 5.7 traz um sumário do desempenho das soluções encontradas pelo SSFMO e MOGA, relatando apenas a freqüência relativa do número de soluções encontradas melhores que a solução padrão.

Comparando o AGMO com o SSFMO, percebemos que o AGMO encontra um número maior de soluções melhores que a padrão em 8 casos, enquanto o SSFMO é melhor nos outros 10. Isso mostra que uma versão multiobjetiva de um método seqüencial pode obter resultados tão bons quanto ou ainda melhores que os produzidos por um AGMO.

Tabela 5.7: Sumário dos resultados dos experimentos utilizando o AGMO e o SSFMO

	AGMO	SSFMO
Base de Dados	F_{Domina}	F_{Domina}
Arrhythmia	0.21	**0.54**
Balance-Scale	**0.7**	0.5
Bupa	0.31	**0.65**
Car	0.002	**0.07**
Crx	0.56	**0.89**
Dermatology	**0.8**	0
Glass	0	**0.99**
Ionosphere	**0.37**	0.14
Iris	0.8	**0.86**
Mushroom	**0.68**	0
Pima	0.34	**0.95**
Promoters	**0.33**	0.27
Sick- euthyroid	0.02	**0.1**
Tic tac toe	0	**0.11**
Vehicle	**0.25**	0.17
Votes	**0.6**	0.1
Wine	0.48	**0.92**
Wiscosin	**0.5**	0.45

Além dos dois métodos multiobjetivos para seleção de atributos propostos neste trabalho, experimentos com o SPEA foram realizados (ver Seção 3.7.4). O SPEA foi escolhido por seu ótimo desempenho em testes comparativos entre os mais tradicionais algoritmos genéticos multiobjetivos da atualidade [Zitzler 2000]. Os resultados obtidos, também em relação à solução padrão com todos os atributos, são apresentados na Tabela 5.8.

Analisando a Tabela 5.8, concluímos que o SPEA consegue encontrar mais de 50% de soluções que dominam a solução padrão em cinco bases: *balance-scale*, *dermatology*, *iris*, *mushroom* e *votes*. Na base *glass* nenhuma solução encontrada pelo SPEA dominou a solução padrão. *Arrhythmia, glass* e *vehicle* são bases em que o valor de $F_{dominado}$ é maior do que o valor de F_{domina}, o que significa que um número maior de soluções encontradas era pior que a solução padrão.

Como nos experimentos realizados com o AGMO, *car* e *tic-tac-toe* apresentaram um grande número de soluções neutras, próximo a 100% do total.

Comparando os resultados obtidos pelo SPEA com os resultados do AGMO proposto, percebemos que, em geral, o número total de soluções encontradas pelo SPEA é superior ao encontrado pelo AGMO.

Porém, o aumento no número de soluções não afeta diretamente o número de soluções melhores que a padrão encontradas (representadas por F_{domina}). Normalmente,

os valores de F_{domina} encontrados pelo AGMO são superiores aos do SPEA. Esse fato pode ser facilmente observado em 4 das 5 bases em que o SPEA encontra mais de 50% de soluções melhores que a padrão. Nessas 4 bases (*balance-scale*, *dermatology*, *iris* e *mushroom*), o número total de soluções encontradas pelo SPEA é superior ao encontrado pelo MOGA, mas os valores de F_{domina} do AGMO são superiores ao do SPEA em todos os casos.

Tabela 5.8: Resultados dos experimentos utilizando o SPEA

	Soluções geradas pelo SPEA						
Base de Dados	Total	Domina	F_{Domina}	Dominado	$F_{Dominado}$	Neutros	$F_{Neutros}$
Arrhythmia	3.0 ± 0.33	1.1 ± 0.34	0.37	1.2 ± 0.36	**0.4**	0.7 ± 0.3	0.23
Balance-Scale	2.4 ± 0.22	1.3 ± 0.26	**0.54**	0.2 ± 0.13	0.08	0.9 ± 0.28	0.38
Bupa	13.0 ± 1.14	5.1 ± 1.65	0.4	0	0	7.9 ± 1.9	0.6
Car	39.8 ± 0.74	0.3 ± 0.15	0.01	0	0	39.5 ± 0.75	0.99
Crx	12.6 ± 2.61	5.8 ± 1.07	0.46	1.2 ± 0.73	0.1	5.6 ± 2.1	0.44
Dermatology	3.3 ± 0.67	2.4 ± 0.54	**0.73**	0	0	0.9 ± 0.38	0.27
Glass	28.7 ± 6.01	0	0	2.9 ± 2.68	**0.1**	25.8 ± 5.79	0.9
Ionosphere	3.9 ± 0.64	1.1 ± 0.38	0.28	0.7 ± 0.33	0.18	2.1 ± 0.41	0.54
Iris	5.8 ± 0.68	4.1 ± 0.99	**0.71**	0.1 ± 0.1	0.01	1.6 ± 0.85	0.28
Mushroom	3.3 ± 0.49	1.7 ± 0.47	**0.52**	0.4 ± 0.22	0.12	1.2 ± 0.29	0.36
Pima	11.6 ± 1.63	3.1 ± 0.96	0.27	0.1 ± 0.1	0.01	8.4 ± 1.8	0.73
Promoters	8.8 ± 2.8	3.1 ± 1.22	0.35	0.6 ± 0.4	0.07	5.1 ± 1.92	0.58
Sick- euthyroid	29.3 ± 1.81	1.4 ± 0.65	0.05	0.6 ± 0.31	0.02	27.3 ± 1.98	0.93
Tic tac toe	20.8 ± 1.15	0.1 ± 0.1	0.01	0	0	20.7 ± 1.21	0.99
Vehicle	4.9 ± 0.48	0.4 ± 0.16	0.08	1.7 ± 0.36	**0.35**	2.8 ± 0.68	0.57
Votes	25.3 ± 3.76	14.7 ± 4.55	**0.58**	1.0 ± 0.73	0.04	9.6 ± 3.67	0.38
Wine	10.7 ± 1.9	2.2 ± 1.2	0.2	1.8 ± 0.77	0.17	6.7 ± 1.42	0.63
Wiscosin	15.9 ± 2.7	5.8 ± 2.26	0.37	1.2 ± 0.42	0.07	8.9 ± 2.36	0.56

Em duas bases, *glass* e *pima*, o número de soluções encontradas pelo AGMO foi superior ao encontrado pelo SPEA. Em *pima*, esse número indica também um valor de F_{domina} maior. Em *glass*, ele afeta apenas o número de soluções neutras.

Quanto ao número de bases em que o número de soluções ruins encontradas é superior ao número de soluções boas, a grande surpresa é a base *vehicle*. Enquanto o AGMO encontra 25% de soluções superiores à solução padrão e 18% piores, no SPEA esses valores são, respectivamente, 8% e 35%.

Em *tic-tac-toe*, por sua vez, o SPEA conseguiu encontrar pelo menos 1% de soluções melhores que a padrão, enquanto no AGMO nenhuma solução melhor que a padrão foi encontrada.

Sumarizando, ao comparamos os valores de F_{domina} para o AGMO (Tabela 5.5) com os valores de F_{domina} para o SPEA (Tabela 5.8), observamos que o AGMO ganha do SPEA em 11 bases, enquanto que o inverso ocorre em 7 bases de dados.

Assim, podemos concluir que o AGMO proposto apresenta resultados um pouco melhores que os apresentados pelo SPEA.

5.2.1 Avaliação das soluções encontradas

Conforme mencionado anteriormente, avaliar a qualidade das soluções geradas por um algoritmo de otimização multiobjetivo é uma tarefa bastante difícil. Idealmente, qualquer tipo de avaliação deve sempre considerar três pontos [Zitzler e Thiele 1999]:

1. A distância do conjunto de soluções não dominadas obtidas do conjunto de soluções ótimas (conjunto de Pareto)
2. A distribuição das soluções encontradas em relação ao conjunto de Pareto
3. O número de soluções encontradas pertencentes ao conjunto de Pareto ótimo.

Porém, no caso da tarefa de seleção de atributos, abordada neste trabalho, não existe um fronte de Pareto ótimo, pois não existe uma função de classificação correta conhecida, como na maioria dos problemas do mundo real. As soluções encontradas são dependentes dos dados disponibilizados ao classificador, e a distribuição ideal dessas soluções é desconhecida.

Assim, a avaliação apresentada anteriormente se concentrou em comparar a qualidade das soluções encontradas pelo AGMO e pelo SSFMO com a solução padrão, com base no conceito de dominância de Pareto.

Em todo caso, para complementar aquela avaliação, também foram analisados alguns aspectos das distribuições de soluções ao longo do fronte de Pareto estimado pelo algoritmo.

As Figuras 5.1 a 5.18 mostram as soluções encontradas pelo AGMO e pelo SSFMO. Os experimentos que retornaram essas soluções foram realizados utilizando 100% dos dados disponíveis, sem distinção entre partição de treinamento e teste. Note que esses experimentos foram realizados para avaliar a diversidade das soluções

encontradas, uma vez que não faz sentido calcular a média das soluções encontradas em 10 partições da validação cruzada.

Analisando os gráficos das Figuras 5.1, 5.5, 5.12 e 5.16, pode-se notar que em *arrhythmia*, *crx, promoters* e *votes*, as soluções encontradas pelo SSFMO estão concentradas em uma pequena região do espaço, enquanto as soluções geradas pelo AGMO encontram-se mais bem distribuídas. Do ponto de vista multiobjetivo, quanto mais distribuídas as soluções estiverem no espaço e ao longo do fronte de Pareto, mais eficiente o algoritmo. Porém, se analisarmos a Tabela 5.7, percebemos que os resultados do SSFMO em a*rrhythmia*, por exemplo, é de 54% de soluções melhores que a solução padrão com todos os atributos, enquanto que o AGMO encontrou apenas 20% dessas soluções, apesar de sua melhor distribuição. O mesmo caso de a*rrhythmia* ocorre em *crx*, onde 89% das soluções são melhores que a padrão com o MOSSF e 56% com o AGMO. Já em *promoters* o SSFMO retorna apenas uma solução, e em *votes* o SSFMO apresenta 10% de soluções melhores que a padrão contra 60% do AGMO.

As Figuras 5.6 e 5.15 mostram que nas bases *dermatology* e *vehicle*, as soluções geradas pelo SSFMO encontram-se do lado direito do gráfico, e as encontradas pelo AGMO do lado esquerdo. Isso significa que o SSFMO encontrou um maior número de soluções com taxas de erro grandes e tamanhos de árvores pequenos, e o AG com taxas de erro pequenas e tamanho de árvores maiores. Analisando então a Tabela 5.7, observa-se que o AG encontrou 80% de soluções melhores que a padrão para *dermatology*, enquanto o SSFMO não encontrou nenhuma. Nessas duas bases o número de soluções neutras predomina.

Já o AGMO concentra soluções em apenas duas regiões do espaço para três bases: *glass, mushroom* e *wiscosin*, como mostram as Figuras 5.7, 5.10 e 5.18. Observando as figuras, as soluções geradas pelo AGMO, nos dois primeiros casos, localizam-se no eixo x, ou seja, possuem erro 0 e seu tamanho varia no eixo x. G*lass* e *mushroom* possuem taxas de erros pequenas, quando executadas com todos os atributos utilizando o C4.5. Para *mushroom* a taxa de erro é 0 e o tamanho da árvore 32.7, e para *glass* a taxa de erro 1.86 e o tamanho da árvore 11. Em *mushroom,* o AGMO encontra 68% de soluções melhores que a padrão, e o SSFMO 100% de soluções neutras. Em *glass*, a situação se inverte: o SSFMO encontra 99% de soluções melhores que a padrão, e o AGMO 94% de soluções neutras. *Wiscosin* apresenta 50% de soluções melhores que a padrão, apesar de suas soluções estarem concentradas em uma só região do espaço.

Pode-se concluir que tanto o AGMO quanto o SSFMO, em alguns casos, não conseguiram obter um conjunto de soluções bem distribuído ao longo do fronte de Pareto. Entretanto, as bases com muitas soluções concentradas, na maioria dos casos, apresentaram números elevados de soluções melhores que a padrão ou de soluções neutras.

Como já afirmado anteriormente, num caso de classificação, a área explorável do espaço de busca é fortemente dependente dos dados apresentados. Na maioria das bases utilizadas nos experimentos, não houve concentração de soluções em apenas uma região do espaço. O AGMO proposto conseguiu, na maioria dos casos, além de obter soluções melhores que a padrão, obter soluções pertencentes a diversas regiões do espaço de busca.

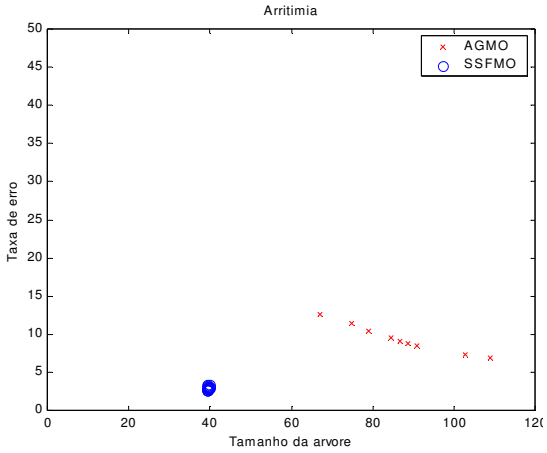


Figura 5.1: Soluções encontradas para Arrhythmia

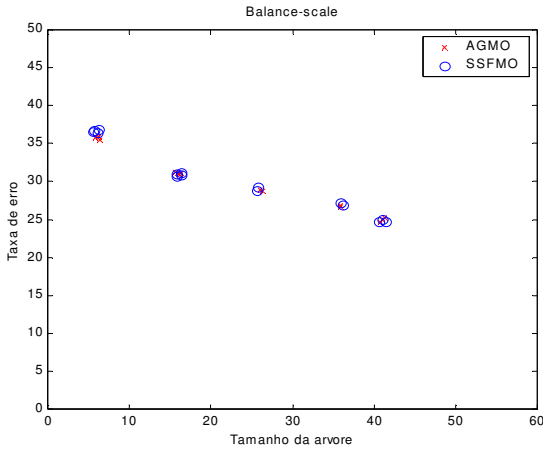


Figura 5.2: Soluções encontradas para balance-scale

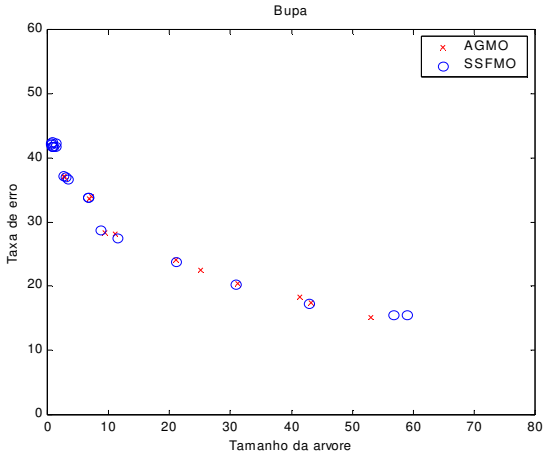


Figura 5.3: Soluções encontradas para Bupa

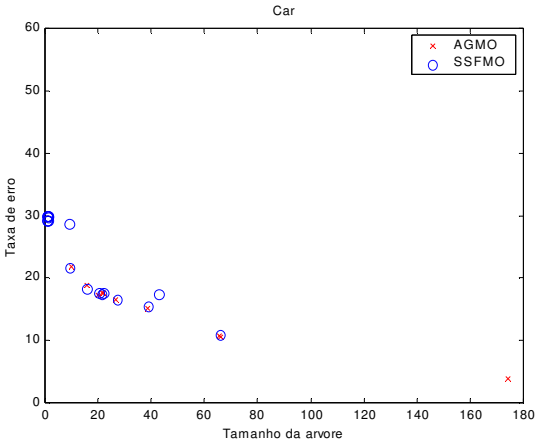


Figura 5.4: Soluções encontradas para Car

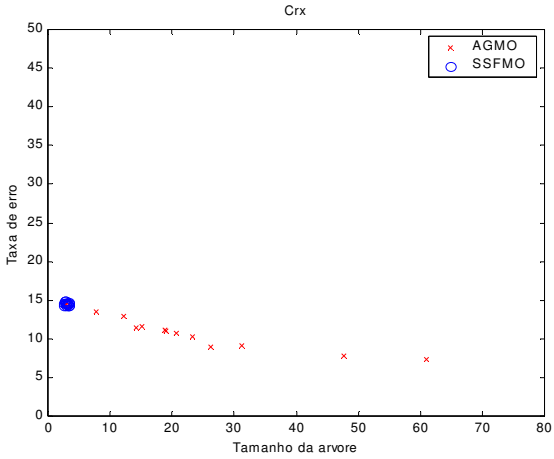


Figura 5.5: Soluções encontradas para Crx

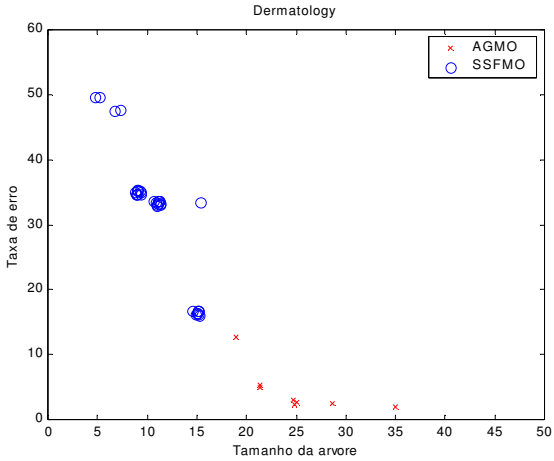


Figura 5.6: Soluções encontradas para Dermatology

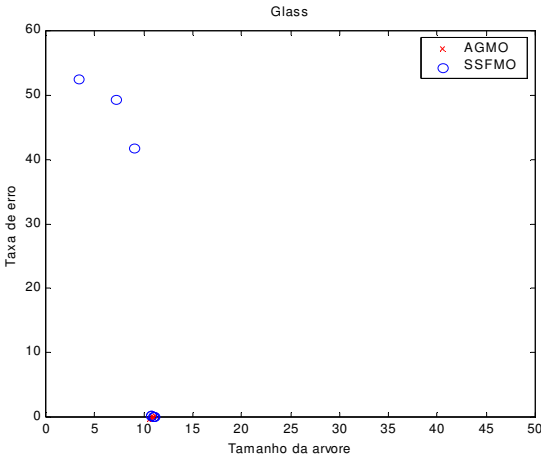


Figura 5.7: Soluções encontradas para Glass

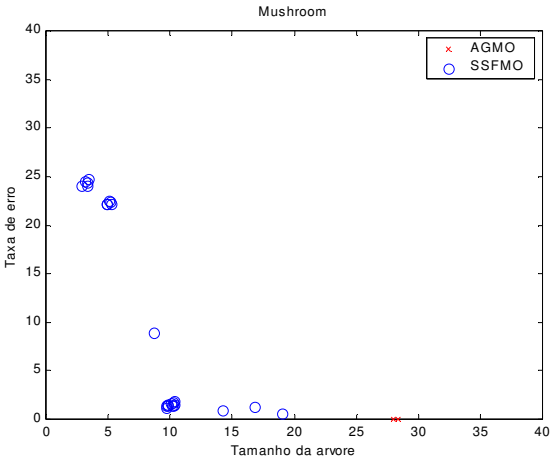


Figura 5.10: Soluções encontradas para Mushroom

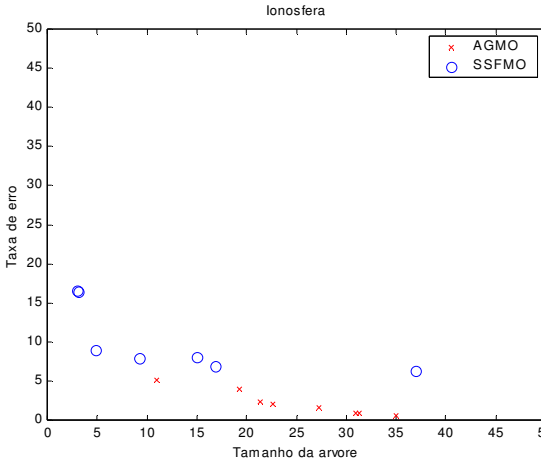


Figura 5.8: Soluções encontradas para Ionosphere

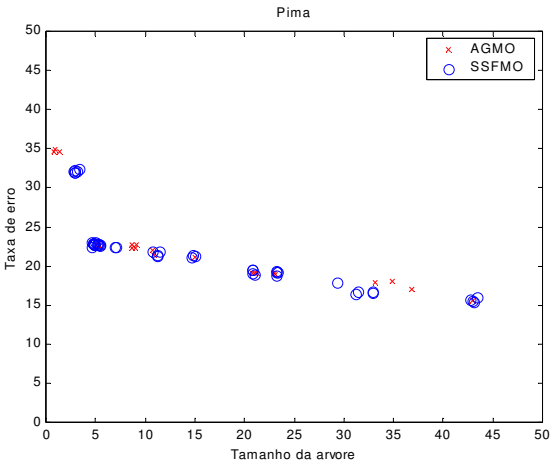


Figura 5.11: Soluções encontradas para Pima

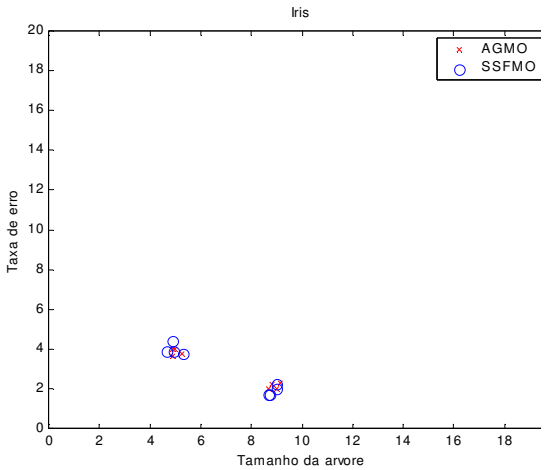


Figura 5.9: Soluções encontradas para Iris

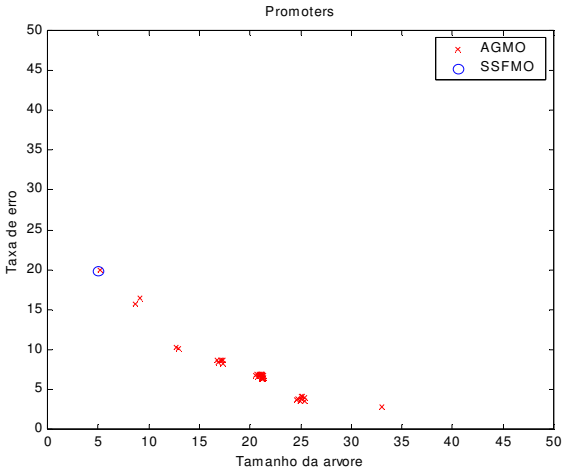


Figura 5.12: Soluções encontradas para Promoters

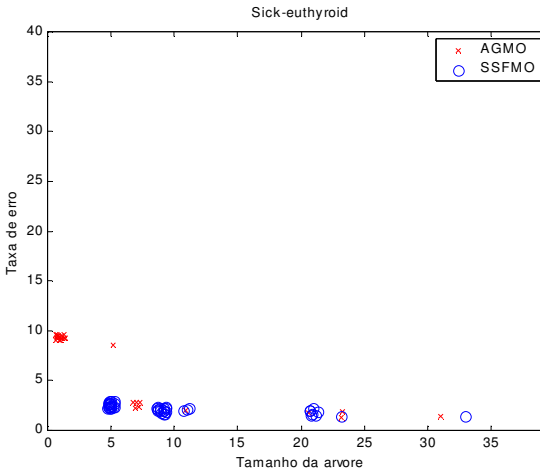


Figura 5.13: Soluções encontradas para sick-euthyroid

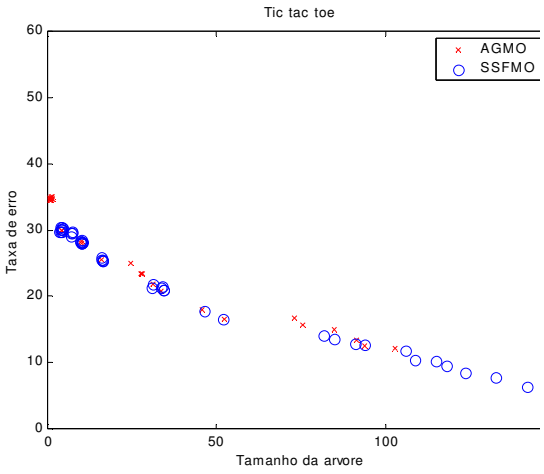


Figura 5.14: Soluções encontradas para Tic-tac-toe

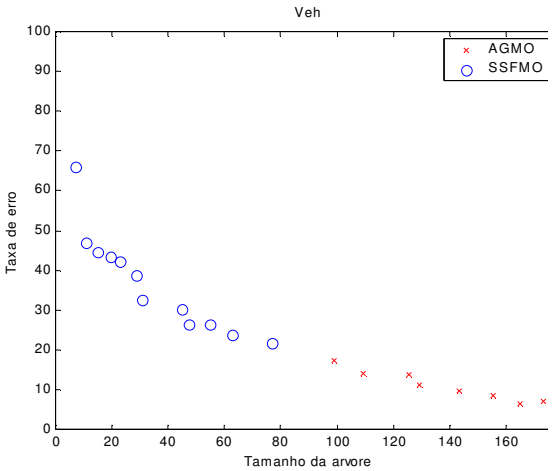


Figura 5.15: Soluções encontradas para Veh

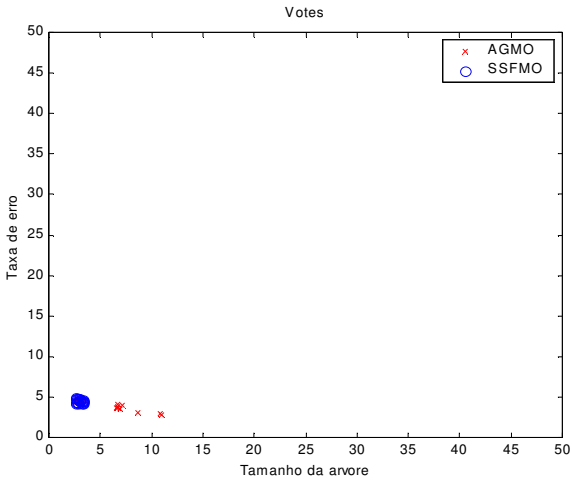


Figura 5.16: Soluções encontradas para Votes

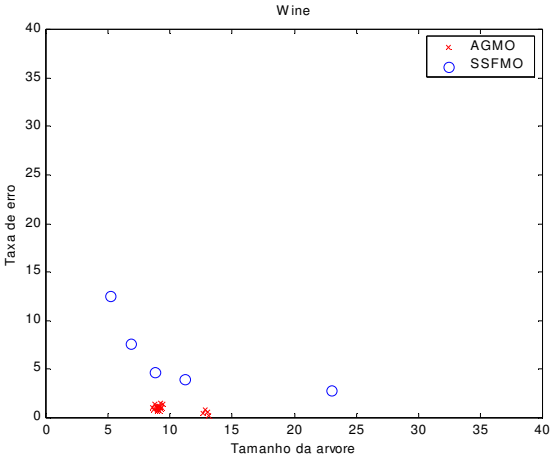


Figura 5.17: Soluções encontradas para Wine

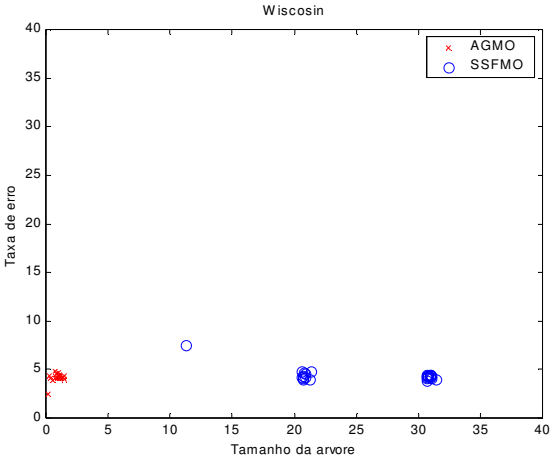


Figura 5.18: Soluções encontradas para Wiscosin

6. TRABALHOS RELACIONADOS

Existem vários trabalhos na literatura que utilizam algoritmos genéticos para a seleção de atributos, a maioria deles seguindo a abordagem *wrapper*. Em [Martín-Bautista e Vila 1999] pode-se encontrar uma revisão dos diversos métodos propostos para solução do problema de seleção de atributos utilizando algoritmos genéticos.

Analisando esses trabalhos, percebe-se que a maioria dos métodos de seleção de atributos, utilizando algoritmos genéticos, pretende resolver problemas de otimização multiobjetiva.

Porém, a maioria deles ainda não utiliza a abordagem ideal definida na Figura 2.6, e sim a abordagem baseada na preferência definida na Figura 2.7.

Em [Chekauer e Shavlik 1996], por exemplo, apresenta-se o SET-Gen, um algoritmo genético para seleção de atributos que visa melhorar a compreensibilidade (reduzir o tamanho) das árvores de decisão geradas pelo C4.5, sem prejudicar a precisão preditiva do classificador.

No SET-Gen, cada indivíduo é representado por um vetor de tamanho fixo, onde cada gene pode conter um atributo ou estar vazio. Um atributo pode ocorrer várias vezes em qualquer posição. Essa codificação foi adotada por tornar mais lenta a perda da diversidade da população, que tende a ocorrer durante a busca genética, e permite que os atributos melhores se proliferem. Além disso, nesse esquema de codificação, o número de genes independe do número de atributos presentes na tarefa de classificação.

A função de avaliação é calculada utilizando a abordagem *wrapper*, e o classificador em questão é o C4.5. São consideradas no cálculo da função de avaliação as seguintes medidas: número de atributos presentes na solução candidata (F), tamanho médio das árvores (S) e a média das taxas de classificação (A) geradas durante a validação cruzada feita sobre os dados de treinamento.

Por utilizar um esquema de otimização multiobjetiva por preferência, a função de avaliação é definida como na Fórmula 6.1.

$$Função\ de\ Avaliação = \frac{3}{4}A + \frac{1}{4}\left(1 - \frac{S+F}{2}\right) \qquad (6.1)$$

Como o objetivo desse trabalho é aumentar a compreensibilidade do conhecimento gerado pela árvore, sem diminuir a taxa de classificações corretas, um peso maior foi definido para A.

O método implementado utiliza os operadores tradicionais de mutação e cruzamento, e introduz um novo operador denominado remoção de atributos. Esse operador retira de um indivíduo todas as ocorrências de um determinado atributo, produzindo um novo indivíduo com um atributo selecionado a menos que o indivíduo original.

Os experimentos mostraram que o SET-Gen diminuiu consideravelmente a complexidade das árvores induzidas, se comparada com o C4.5, tanto em relação ao tamanho quanto ao número de atributos referenciados. Porém, reduções significativas na taxa de classificações corretas do C4.5 não foram encontradas.

Em [Bala *et* al 1996], a seleção de atributos é realizada através de um algoritmo genético denominado GENESIS 4.5, com operadores e parâmetros padrão da literatura. Tratando-se da função de avaliação dos indivíduos, utilizou-se uma abordagem *wrapper*+filtro.

Segundo os autores, normalmente preocupa-se muito com a taxa de classificações corretas obtida e ignora-se a necessidade de minimizar o número de atributos utilizados para classificação. Para resolver esse problema, a abordagem proposta avalia um subconjunto de atributos selecionados de 3 formas: pela precisão preditiva obtida pelo C4.5 e por outras duas medidas heurísticas denominadas conteúdo da informação (estimado através da teoria da informação) e custo (leva em consideração o número de atributos presentes na solução candidata). O primeiro critério é baseado na abordagem *wrapper*, pois é obtido da saída do C4.5. O segundo e o terceiro são baseados na abordagem filtro.

Partindo dessas três medidas, a função de avaliação descrita na Fórmula foi criada:

$$Função\ de\ Avaliação = Informação - Custo + Precisao\_Preditiva \qquad (6.2)$$

Para avaliar o desempenho do método, ele foi comparado com C4.5 e C4.5 + abordagem filtro, e obteve resultados com subconjuntos de atributos pequenos e taxa de classificações corretas iguais ou melhores que a do C4.5.

Muitos outros métodos para seleção de atributos utilizam algoritmos genéticos multiobjetivos baseados em preferência, como em [Ishibuchi e Nakashima 2000] e [Wang e Sundaresh 1998].

Porém, apesar da simplicidade da otimização multiobjetiva baseada em preferência, outros trabalhos recentes estão utilizando a definição ideal de otimização multiobjetiva para resolver problemas de seleção de atributos com mais de um objetivo.

[Emmanouilidis *et* al 2000] propõe um algoritmo genético para seleção de atributos, utilizando conceitos ideais de otimização multiobjetiva, baseado no NPGA (*Niched-Pareto Genetic A*lgorithm), e justifica seu uso por evitar que restrições *a priori* sejam impostas à busca. O algoritmo genético implementado possui as seguintes características:

- A população de novos indivíduos (população filha) possui sempre o dobro de indivíduos da população atual (população pai).
- De cada população é retirado um conjunto elitista, formado pelos melhores indivíduos da população. O tamanho da população pode ser alterado para garantir que seja pelo menos 10 vezes maior que o número de indivíduos pertencente ao conjunto elitista.
- A função de avaliação é calculada utilizando a abordagem *wrapper*, e o classificador considerado é uma rede neural, que pode ser probabilística ou um perceptron multicamadas. Três objetivos são considerados: a taxa de erro, o tamanho do conjunto de atributos selecionados e uma função de custo, utilizada apenas para comparar indivíduos com a mesma taxa de erro.
- O diferencial desse algoritmo está nos operadores utilizados. A seleção é a única realizada como nos métodos tradicionais, mas utiliza *fitness sharing*.

  A mutação é adaptativa. Quando progresso é observado em uma geração, com a inclusão de novos indivíduos não-dominados no conjunto elitista, por exemplo, a taxa de mutação decresce. Quando nenhum progresso é observado, ela é acrescida.

  O operador de cruzamento é baseado em associação e ajuda a preservar blocos construídos com precisão preditiva promissora. Seu conceito baseia-se na idéia de que operadores convencionais tendem a gerar indivíduos em que o número de atributos selecionados é a média do número de atributos selecionados nos pais, dando prioridade a conjuntos de atributos de tamanhos médios. No operador baseado em associação, os indivíduos têm em média o mesmo número de atributos dos pais, permitindo assim que a diversidade da população seja mantida também no tamanho do subconjunto de atributos.

Quando os resultados foram comparados com a busca seqüencial, observou-se que o método proposto encontra muitas soluções perdidas pela busca seqüencial, além de identificar soluções tão boas quanto as identificadas por ela.

Além disso, os experimentos mostraram também que, nas bases de dados utilizadas nos experimentos, as redes neurais probabilísticas foram melhores classificadores que as perceptron multicamadas.

Em [Kim *et* al 2000] uma abordagem um pouco diferente das anteriores é encontrada. Este trabalho resolve a tarefa de seleção de atributos como pré-processamento para tarefa de agrupamento (*clustering*), em vez da tarefa de classificação. Existem vários critérios heurísticos que podem ser utilizados para estimar a qualidade de agrupamentos construídos a partir de um subconjunto de atributos. Esse método combina essas heurísticas com um algoritmo evolucionário chamado ELSA (*Evolutionary Local Search Algorithm*), baseado em seleção local e que mantém uma população diversa de soluções que se aproximam do fronte de Pareto, num espaço multiobjetivo.

A abordagem *wrapper* também está presente nesse método, e o algoritmo K-média (*k-means*) é utilizado com o algoritmo de agrupamento.

Nesse caso, o algoritmo evolucionário é utilizado tanto para pesquisar o espaço de busca, quanto para determinar o número de agrupamentos *k* que poderão ser formados pelo K-média, já que o valor de k faz parte do material genético do indivíduo.

ELSA é um algoritmo evolucionário, baseado em modelos de vida artificial de agentes adaptativos em ambiente ecológico, e utiliza alguns conceitos um pouco diferentes dos de algoritmos genéticos. Ele também utiliza uma função de avaliação e o operador de mutação, mas não utiliza operador de cruzamento.

Foram utilizados 4 critérios heurísticos como função de avaliação, o primeiro considerando a coesão dos agrupamentos, o segundo favorecendo os agrupamentos bem separados, e os dois últimos baseados na navalha de Occam [Domingos 1998]: modelos com poucos agrupamentos e poucos atributos selecionados são mais compreensíveis e, portanto, são favorecidos.

Experimentos foram realizados inicialmente com bases de dados artificiais, e posteriormente com a base Wisconsin Breast Cancer [Murphy e Aha 1994]. A análise dos resultados foi feita de acordo com a relevância clínica dos agrupamentos resultantes e, segundo os autores, apresentou resultados satisfatórios.

Em relação a esses trabalhos, o método aqui propondo partilha de algumas idéias. Ele utiliza a abordagem *wrapper* para cálculo da função de avaliação, como todos os métodos discutidos acima, e alguns objetivos a serem otimizados são comuns. [Chekauer e Shavlik 1996] utilizam a taxa de acerto, o tamanho da árvore e o número de atributos selecionados, [Bala *et* al 1996] a taxa de acerto e duas outras heurísticas, incluindo o número de atributos selecionados, [Emmanouilidis *et* al 2000] a taxa de erro, o número de atributos selecionados e em alguns casos uma função de custo, e [Kim *et* al 2000] utiliza apenas medidas heurísticas, mas considera o número de atributos selecionados.

O método proposto neste trabalho utiliza a taxa de acerto e o tamanho da árvore gerada pelo classificador, e um terceiro objetivo com o mesmo propósito (porém de natureza bem diferente) da função de custo de [Emmanouilidis *et* al 2000]. Uma medida encontrada em todos os métodos anteriormente citados e que não aparece na abordagem proposta é o número de atributos selecionados. Isso ocorre porque a medida do tamanho da árvore gerada já é um bom indicador para a compreensibilidade das regras geradas, além de ser uma medida mais direta. Considere, por exemplo, um caso em que um subconjunto de 10 atributos gera uma árvore com 16 nós e taxa de erro de 5%. Se desse subconjunto de 10 atributos um for retirado, restando 9 atributos, pode-se obter uma árvore com 30 nós e mesma taxa de erro de 5%. Isso porque o atributo retirado era essencial para construção da árvore mais compacta. Portanto, nesse trabalho o tamanho da árvore tende a ser uma medida mais eficaz e direta.

Quanto aos operadores, o método proposto utiliza os tradicionais da literatura, mas fica a possibilidade de, no futuro, utilizar o operador de cruzamento de [Emmanouilidis *et* al 2000], para analisar seus efeitos no algoritmo genético proposto.

O método proposto neste trabalho utiliza, parar desempatar um torneio, um critério definido como F_3, descrito na Seção 4.1.2. Esse critério leva em consideração tanto o número de indivíduos dominados por um indivíduo *I* quanto os indivíduos que dominam o indivíduo *I.* Esse critério é uma extensão do critério utilizado na literatura, que considera apenas os indivíduos que *I* domina.

Finalmente, este trabalho propôs um algoritmo de seleção seqüencial para a frente multiobjetivo, o qual foi comparado com o algoritmo genético multiobjetivo. Isso não foi realizado em nenhum outro trabalho da literatura, até onde se estende o conhecimento da autora.

7. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho propôs um algoritmo genético multiobjetivo para seleção de atributos em tarefas de classificação e uma versão multiobjetiva do método de seleção seqüencial para frente, tradicionalmente utilizado na literatura como método de seleção de atributos.

Seu principal objetivo era observar o comportamento de versões multiobjetivas de métodos que vêm sendo utilizados há muito tempo na tarefa de seleção de atributos, porém considerando-a como uma tarefa de otimização de apenas um objetivo (ou combinando alguns objetivos em funções ponderadas, como mostrado em [Bala *et* al 1996], [Ishibuchi e Nakashima 2000] e [Wang e Sundaresh 1998]).

Um dos desafios desse trabalho foi definir qual(is) solução(ões) seriam retornadas pelo AGMO (uma vez que não existia um usuário para escolher a melhor solução, como dita a otimização multiobjetiva) e como essas soluções retornadas pelo AGMO seriam então avaliadas como boas ou ruins.

Depois de definir o C4.5 como base de comparação para as soluções encontradas pelos métodos propostos, os experimentos realizados mostraram que o AGMO proposto conseguiu encontrar soluções melhores que a padrão (consistindo de todos os atributos) em 16 das 18 bases da UCI [Murphy e Aha 1994] utilizadas nos experimentos, resultados muito superiores aos encontrados pelo AG simples, que só conseguiu soluções com taxas de erro e tamanhos de árvore menores que a solução padrão em *balance-scale*.

O AGMO proposto foi também aplicado a uma base de dados de produtos Motorola, composta por uma média de 15.000 registros distribuídos em 2 classes e descritos por 109 atributos. O método selecionou 27 desses 109 atributos, com taxas de erro e tamanhos de árvore obtidos menores que o da solução padrão (solução com todos os atributos) executada com o C4.5.

O AGMO proposto foi também comparado com o SPEA [Zitzler e Thiele 1999], considerado um dos algoritmos mais eficazes para resolver problemas multiobjetivos da atualidade.

De modo geral, os resultados do AGMO foram um pouco melhores que os do SPEA.

Os resultados mais surpreendentes foram os encontrados pelo SSFMO, que mostrou poder competir tranqüilamente com o AGMO. Comparando os dois métodos, em relação ao número de soluções encontradas consideradas melhores que a padrão, o AGMO é melhor que o SSFMO em 8 bases, e esse último melhor que o AGMO em 10 bases.

Portanto, os resultados deste trabalho fornecem evidência significativa que métodos de busca seqüenciais, no caso da tarefa de seleção de atributos, podem também ser utilizados com conceitos multiobjetivos com eficácia e eficiência.

Tanto o AGMO quanto o SSFMO mostraram-se como boas alternativas para a solução do problema de seleção de atributos.

Como trabalhos futuros, uma análise das características das bases de dados onde os métodos propostos apresentaram seus melhores resultados deve ser realizada, visando encontrar padrões capazes de descrever as bases em que os métodos podem ser aplicados com sucesso.

Um projeto para que o método retorne apenas uma solução dentre as encontradas pelo AGMO também deve ser implementado, resolvendo assim o problema encontrado em sistemas em que não existe um usuário capaz de escolher uma dentre as soluções retornadas.

Além disso, pode-se aumentar o número de objetivos a serem otimizados, incluindo na avaliação, por exemplo, o número de atributos selecionados.

Pretende-se também avaliar o desempenho do algoritmo genético multiobjetivo proposto nesse trabalho, utilizando uma abordagem filtro na atribuição da função de avaliação. A abordagem filtro oferece grande vantagem em relação a *wrapper* quanto ao custo computacional. A mesma proposta pode ser estendida à versão multiobjetiva do método de seleção seqüencial para frente.

Pode-se também incluir aos objetivos a serem otimizados essas mesmas métricas utilizadas em abordagens filtro, como a taxa de inconsistência e o tamanho dos dados, como utilizado em [Wang e Sundaresh 1998].

Além disso, as vantagens da otimização multiobjetiva podem ser examinadas também em outros métodos de busca seqüencial, como a seleção seqüencial para trás.

Outros métodos de *niching* podem ser também incorporados ao AGMO proposto, buscando espalhar ainda mais as soluções em torno do espaço de soluções e do fronte de Pareto.

Além disso, métricas de avaliação e comparação para algoritmos multiobjetivos devem ser estudadas e posteriormente aplicadas para uma comparação mais formal dos dois algoritmos aqui propostos.

REFERÊNCIAS

[Aha e Bankert 1996] Aha, David W.; Bankert, Richard L., A Comparative Evaluation of Sequential Feature Selection Algorithms, Learning from Data: AI and Statistics V, Eds: Fisher, D. e Lenz, H. J., Springer-Verlag, 1996.

[Bala et al 1996] Bala, J.; Jong K. De; Huang, J.; Vafaie, H.; Wechsler, H; Using Learnig to Facilite the Evolution of Features for Recognizing Visual Concepts, In: Special Issue of Evolutionary Computatuion – Evolution, learning and Instinct: 100 years of Baldwin Effect, Vol. 4 , pp. 297-311. 1996.

[Bhattacharyya 2000] Bhattacharyya, S., Evolutionary Algorithms in Data mining: Multi-Objective Performance Modeling for Direct Marketing, In: Proc. KDD-2000, pp 465-471, 2000.

[Blum e Langley 1997] Blum, A. L.; Langley, P., Selection of Relevant Features and Examples in Machine Learning, In: Special Issue of Artificial Intelligence on Relevance, Greiner, R. e Sbramanian (Eds), pp 245-271, 1997.

[Boz 2002] Boz, O., Feature Subset Selection by Using Sorted Feature Relevance, In: ICMLA 2002 – International Conference on Machine Learning and Applications, USA, 2002.

[Caruana e Freitag 1994] Caruana, R.; Freitag, D., Greedy Attribute Selection, In: Proc of 11th International Conference in Machine Learning, pp 28-36, 1994.

[Chekauer e Shavlik 1996] Chekauer, K. J.; Shavlik, J. W., Growing Simple Decision Trees to Facilite Knowledege Discovery, In: KDD-1996, pp 315-318, 1996.

[Coello et al 2002] Coello, C. A. C.; Veldhuizen, D. A. V. e Lamont, G. B., Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, Nova Iorque, 2002.

[Deb e Goldberg 1989] Deb, K.; Goldberg, D. E., An investigation of Niche and Species Formation in Genetic Function Optimization, In: Proc. of International Conference on Genetic Algorithms, pp 42-50, 1989.

[Deb 1999] Deb, K, Multi-Objective Evolutionary Algorithms: Introducing Bias Among Pareto-Optimal Solutions, Kanpur Genetic Algorithms Laboratory Report nº 99002, India,1999.

[Deb 2000] Deb, K., Introducion to Selection, In:Back, T.; Fogel, D.B.; Michalewicz (Eds.), Evolutionary Computation 1, Institute of Physics Publishing, pp 166-171, 2000.

[Deb 2001] Deb, K., Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Inglaterra, 2001.

[Dhar e Stein 1997] Dhar, V.; Stein, R., Seven Methods for Transforming Corporative Data into Business Intelligence, Prentice Hall, Londres, 1997.

[Domingos 1998] Domingos, P., Occam's two Razors: The Sharp and the Blunt, In: Proc. Fourth International Conference on Knowledge Discovery and Data Mining, pp 37-43, 1998.

[Emmanouilidis et al 2000] Emmanouilidis, C.; Hunter, A.; MacIntyre, J., A Multiobjective Evolutionary Setting for Feature Selection and a Commonality-Based Crossover Operator, In: Proc. CEC-2000, pp 301-216, 2000.

[Eshelman et al 1989] Eshelman, L. J.; Caruana, R.A.; Schaffer, J.D.; Biases in the crossover landscape, In: Proc of International Conference on Genetic Algorithms, pp 10-18, 1989.

[Falkenauer 1999] Falkenauer, E., The worth of uniform, In: Proc. IEEE Congress on Evolutionary Computation, pp 776-782, 1999.

[Fayyad et al 1996] Fayyad, U.M.; Piatitsky-Shapiro, G.; Smyth, P.; Uthurusamy, R. (Eds), American Association for Artificial Intelligence, Inglaterra, 1996.

[Fidelis et al 2000] Fidelis, M.V.; Lopes, H.S.; Freitas, A.A., Discovering Comprehensible Classification Rules with a Genetic Algorithm, Proc. Congress on Evolutionary Computation, 2000.

[Fonseca e Fleming 1993] Fonseca, C.M.; Fleming, P.J., Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, In: Proc of 5th Conference on Genetic Algorithms, pp 416-423, 1993.

[Fonseca e Fleming 1994] Fonseca, C.M.; Fleming, P.J.,An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation 3, pp 1-16, 1994.

[Fonseca e Fleming 2000] Fonseca, C.M.; Fleming, P.J., Multiobjetive Optimization; In: Back, T.; Fogel, G.B.; Michalewicz Z. (Eds), Evolutionary Computation 2: Advanced Algorithms and Operators, Institue of Physics Publishing, pp 25-37, 2000.

[Freitas e Lavington 1998] Freitas, A.A.; Lavington, S. H., Mining Very Large Databases with parallel Processing, Kluwer, 1998.

[Freitas 2001] Freitas, A. A.; Understanding the Crucial Role of Attributes Interaction in Data Mining, In: Artificial Intelligence Review 16, pp 177-199, Kluwer Academic Publishers, 2001.

[Freitas 2002] Freitas, A. A., Data Mining and Knowledge Discovery with Evolutionary Algorithms, Berlin: Springer-Verlag, 2002.

[Goldberg 1989] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, 1989.

[Hand 1997] Hand, D. J., Construction and Assessment of Classification Rules, Wiley, Nova Iorque, 1997.

[Hinterding 2000] Hinterding, R., Representation, Mutation and Crossover Issues in Evolutionary Computation, In: Proc. Conference on Evolutionary Computation, pp 916-923, 2000.

[Holsheimer e Siebes 1991] Holsheimer, M.; Siebes, A., Data Mining – The Search for Knowledge in Databases, Report CS-R9406, Amsterdam, 1991.

[Horn et al 1994] Horn, J.; Nafpliotis, N.; Goldberg, D. E., A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In Proceedings of the ICEC '94, IEEE, pp. 82-87, 1994.

[Ishibuchi e Nakashima 2000] Ishibuchi, H.; Nakashima, T., Multi-objective pattern and feature selection by a genetic algorithm, In: Proc. GECCO–2000, pp 1069-1076, 2000.

[John et al 1994] John, G. H.; Kohavi, R.; Pfleger, K., Irrelevant Features and the Subset Selection Problem, 11th International Conference in Machine Learning, pp 121-129, 1994.

[Kim et al 2000] Kim,Y.; Street, W. N.; Menczer, F., Feature Selection in Unsupervisioned Learning via Evolutionary Search, In: Proc. KDD-2000, pp 365-369, 2000.

[Kira e Rendell 1992] Kira, K.; Rendell, L. A., The Feature Selection Problem : Traditional Methods and a New Algorithm, In: Proc. 10th Conference on Artificial Intelligence, pp129-136, Menlo Park, CA, 1992.

[Knowles e Corne 2002] Knowles, J.; Corne, D., On metrics for comparing Nondominated Sets, In: Proc of CEC 2002, pp 711-716, 2002.

[Kohavi e John 1998] Kohavi, R.; John, G. H., The Wrapper Approach, In: H. Liu & H. Motoda (Eds.) Feature Extraction, Construction and Selection: a data mining perspective, 33-49. Kluwer, 1998.

[Koller e Sahami 1996] Koller, D.; Shami, M.; Toward Optimal Feature Selection, In: Proc. Of 13th International Conference in machine Learning, 1996.

[Langley e Iba 1993] Langley, P.; Iba, W., Average-case analysis of a nearest neighbor algorithm, In: Proc. Of 13th International Conference on Artificial Intelligence, pp 889-894, 1993.

[Liu e Motoda 1998] Liu, H.; Motoda, H., Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, 1998.

[Martín-Bautista e Vila 1999] Martín-Bautista, M. J.; Vila, M. A., A Survey of Genetic Feature Selection in Mining Issues, In: Proc. IEEE Conference on Evolutionary Computation, pp 1314-1321, Washington,1999.

[Mathfound 2000] Mathfound, S. W., Niching Methods, In:Back, T.; Fogel, D.B.; Michalewicz (Eds.), Evolutionary Computation 2, pp 87-92, Institute of Physics Publishing, 2000.

[Michalewicz 1996] Michalewicz, Z., Genetic Algorithms + Data Structures : Evolution Programs, Springer, 3rd Ed, Nova Iorque, 1996.

[Murphy e Aha 1994] Murphy, P.M.;Aha, D.W., UCI Repository of Machine Learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of information and Computer Science, 1994.

[Pappa et al 2002a] Pappa, G. L.; Freitas, A. A.; Kaestner, C. A. A., Attribute Selection with a Multiobjective Genetic Algorithm, In: Proc. of 16th Brazilian Symposium on Artificial Intelligence, pp 280-290, 2002.

[Pappa et al 2002b] Pappa, G. L.; Freitas, A. A.; Kaestner, C. A. A., A Multiobjective Genetic Algorithm for Attribute Selection, In: Proc. of 4th International Conference on Research Advances in Soft Computing (RASC), pp 116-121, 2002.

[Srinivas e Deb 1994] Srinivas, N.; Deb, K., Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), pp 221-248, 1994.

[Quinlan 1993] Quinlan, J.R., C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[Vafaie 1993] Vafaie, H.; Jong, K. D., Robust Feature Selection Algorithms, In: Proc. of the 1993 IEEE, Conf. On Tools with AI, pp 356-363, 1993.

[Wang e Sundaresh 1998] Wang, K.; Sundaresh, S., Selecting features by vertical compactness of data. In: H. Liu & H. Motoda (Eds.) Feature Extraction, Construction and Selection: a Data Mining Perspective, pp 71-84. Kluwer, 1998

[Zitzler e Thiele 1999] Zitzler, E.; Thiele, L., Multiobjective Evolutionary Algorithms: A Comparative Study and the Strength Pareto Approach, In: IEEE Transactions on Evolutionary Computation 3(4), pp 257-271, 1999.

[Zitzler et al 2000] Zitzler, E.; Thiele, L.; Deb, K., Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, In: Evolutionary 8(2), pp 173-195, 2000.

[Zitzler et al 2002] Zitzler, E.; Laumanns, M.; Thiele, L.;Fonseca, C. M.; Fonseca, V. G., Why Quality Assessment of Multiobjective Optimizers is Difficult, In: Proc. GECCO–2002, pp 666-673, 2002.