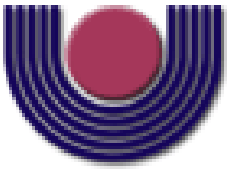


Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Informática
Curso de Bacharelado em Informática

Uma Comparação Experimental entre Algoritmos para Seleção de Características

Rodrigo Matheus da Costa Rodrigues

CASCAVEL
2009

RODRIGO MATHEUS DA COSTA RODRIGUES

UMA COMPARAÇÃO EXPERIMENTAL ENTRE ALGORITMOS PARA SELEÇÃO DE CARACTERÍSTICAS

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Informática, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Clodis Boscarioli

CASCAVEL
2009

RODRIGO MATHEUS DA COSTA RODRIGUES

UMA COMPARAÇÃO EXPERIMENTAL ENTRE ALGORITMOS PARA SELEÇÃO DE CARACTERÍSTICAS

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Informática, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

_____

Prof. Dr. Clodis Boscarioli (Orientador)
Colegiado de Informática, UNIOESTE

_____

Prof. Dr. Jorge Bidarra
Colegiado de Informática, UNIOESTE

_____

Profª. Drª. Cláudia Brandelero Rizzi
Colegiado de Informática, UNIOESTE

Cascavel, 16 de dezembro de 2009

DEDICATÓRIA

Dedico esta monografia a todos que auxiliaram na execução deste texto, entre eles, minha família, meu orientador, minha banca e toda equipe de funcionários, professores, colegas e camaradas que moldaram não apenas aspectos de minha formação acadêmica como também do meu ser.

EPÍGRAFE

Zé

. . .
Queria ser jogador de futebol, jogar na seleção
Fazer um gol de placa no jogo da decisão
Mas sou impedido por um bando de ladrão
Cambada inquisitora que comanda o batalhão
. . .

Zé Ofensa

Oculta na roupagem metafórica palpita a
essência real.

Helena Kolody

Ouro e chumbo

Covardia é o ato de atacar por vencer
ou de fugir quando se pode ganhar
Coragem é lutar para se defender
mesmo quando se pode perder
Um tem valor outro apenas faz peso
Dois metais em um mesmo dispositivo
Em pratos opostos de uma mesma balança

Rodrigo Matheus

AGRADECIMENTOS

Agradeço, primeiramente, a minha família, meu pai Irineo da Costa Rodrigues, minha mãe Vera Lúcia Cavalari e meus irmãos Irineo da Costa Rodrigues Júnior e Marcelo da Costa Rodrigues.

A todos os amigos que fiz até então, desde os amigos em minha cidade natal, Matelândia, até os que fiz nesta cidade durante meus estudos. Alguns deles em ordem cronológica, se isso for possível: Paula, Leandro, Buda, Érica, Daia, André, Daniel, Telma, Pamela, Tuti, Tiago, Scharleston, Bocão, Mauro, Michel, Josué, Andréia, Kiko, Chiba, Rafael, Dolly, Ricardo, Soneca, Josiane, Mauro, Márcio, Pantera, Rafael, Juce, Ane, Polly, Dani, Carla, Pato, Fú, Cabelo, Lucão, Secão, Amanda, Xauz, Cabeça, Ana, Cláudia, Rafa, Goiano, Palhacinho, Zecão, Dárcio, Bereja, Caiçara, Gaúcho, Heremita, Paulo, Marafa, Rodrigo, Bava, Kristopher, Gordinho, Orlando, Thiago, Érica, Portelinha, Alemão, Marrom, Keyse, Mandruvá, Chicão, Chambó, Kadu, Paulinho, Lucas, Bruno, Porco, Grezi, Negão, Micróbio, Maomé, Guigo, Chuchú, Ursinho, Remi, Bel, Ana, Melinda, Pedrão, Leila, Xicó, Quirino, Jamile, Bavaresca, Xaxá, Fat, Érica, Sandy, Pedrão, Bzão, Vivi, Pequeno, Jorge, Jamais, Sensei, Fábio, Fernando, Jeferson, Caio, Lucas, Leonardo, Hiago, Vinícius, Eslone, Érica, Simone, Diego, Marco, João.

A todos os professores, técnicos e funcionários: Aníbal Mantovani Diniz, Ivonei Freitas da Silva, Oscar José Busatta, Victor Francisco Araya Santander, Carlos José Maria Olguin, Adair Santa Catarina, Francisco Sérgio Sambatti, Marcio Seiji Oyamada, Ivan Kuiava, Luiz Antônio Rodrigues, Guilherme Galante, André Luiz Brun, Jacson Renzo Querubin, Suzan Kelly Borges Piovesan, Anete Teresinha Trasel, Elder Elisandro Schemberger, Nelson Raimundo da Rocha, Carin Rosangela Krebs Rendel, Anndri Kappke, Josué Pereira de Castro, Adriana Postal, e ao meu orientador Clodis Boscarioli e aos membros de minha banca, Jorge Bidarra e Cláudia Brandelero Rizzi.

Lista de Figuras

Lista de Algoritmos

Lista de Tabelas

Lista de Abreviaturas e Siglas

AC	Avaliações do Critério
AD	Árvore de Decisão
AP	Acessos aos Padrões
CON	Refere-se ao critério Taxa de Inconsistências
DC	Discretização de Características
EB	Espaço de Busca
EP	Expansões progressivas
ER	Expansões regressivas
PA	Progressão Aritmética
PD	Problema da Dimensionalidade
RD	Redução da Dimensionalidade
SC	Seleção de Características
SFS	*Sequential Forward Search* - Busca Sequencial Progressiva
SBS	*Sequential Backward Search* - Busca Sequencial Regressiva
SFFS	*Sequential Floating Forward Search* - Busca Sequencial Flutuante Progressiva
SFBS	*Sequential Floating Backward Search* - Busca Sequencial Flutuante Regressiva
BAB	*Branch and Bound* - Ramificar e Podar
BAB'	Ramificar e Podar alterado
DEP	*Depth First Search* - Busca em Profundidade
BRD	*Breadth-first Search* - Busca em Amplitude (ou Largura)
PTA	*Plus* l - *Take Away* r - Mais l - Menos r
GPTA	*Generalized Plus* l - *Take Away* r - Mais l - Menos r Generalizado
GSFS	*Generalized Sequential Forward Search* - Busca Sequencial Progressiva Generalizada
GSBS	*Generalized Sequential Backward Search* - Busca Sequencial Regressiva Generalizada
LV	*Las Vegas*
LVF	*Las Vegas Filter* - Filtro LV
LVW	*Las Vegas Wrapper*
LVI	*Las Vegas Incremental*
OS	*Oscillating Search*
ID3	*Iterative Dichotomiser* 3
EWI	*Equal Width Intervals* - Intervalos de Tamanhos Iguais
EFI	*Equal Frequency Intervals* - Intervalos de Frequências Iguais
GPL	*GNU General Public License*
jaDTi	*Decision Trees: a Java implementation*
UDM	(Unioeste *Data Mining*)

Lista de Símbolos

i, j e k	Contadores
v	Um valor
v^{lim}	Um valor limiar
v_i	O i-ésimo valor de uma sequência
l	Fator de *lookahead*, usado em algumas versões do algoritmo BAB
C_N^n	Combinação de N elementos n a n
M	Número de instâncias de dados
N	Quantidade de características, dimensionalidade)
n	Quantidade de características selecionadas
h	Quantidade de características não selecionadas, $h = N - \|S\|$
a	Quantidade de características mínimo para precisão máxima
b	Quantidade de características limiar sem deteriorar a precisão
t	Número de ramos
m	Quantidade de intervalos
q	Quantidade de bases testadas
Υ	Conjunto com todos os padrões de uma base de dados
Υ^{TR}	Conjunto de padrões para treinamento
Υ^{T}	Conjunto de padrões para teste
Υ^{i}	Subconjunto de padrões
$\Upsilon[i]$	Vetor de características, i-ésimo padrão
$\Upsilon[i][j]$	Valor característico, j-ésima característica do i-ésimo padrão
$\|\Upsilon\|$	Quantidade de padrões no subconjunto de padrões
ρ	Padrão, composto pelas características e o alvo
ρ^i	Valor característico da i-ésima característica
λ	Quantidade de classes
$\Lambda(\bullet)$	Obtém a(s) classe(s) associada(s) ao(s) padrão(ões)
c	Classe
c_i	A classe determinada por i
$[c_1, \ldots, c_i]$	Vetor determinado pelos elementos
V	Um vetor qualquer
$\|V\|$	Quantidade de elementos no vetor
$V[i]$	Acessa o i-ésimo elemento do vetor
x	Característica
$\{x_1, \ldots, x_i\}$	Subconjunto determinado pelas características
S	Subconjunto
$\|S\|$	Número de características do subconjunto

S^i	Subconjunto que contém i características
S_*	Subconjunto completo
$S_\oslash$	Subconjunto vazio
$S_\oplus$	Melhor subconjunto local
$S^i_\oplus$	Melhor subconjunto local com i características
$S_\otimes$	Subconjunto ótimo
$S^i_\otimes$	Subconjunto ótimo com i características
$\Psi(\bullet)$	Avaliação do subconjunto
$\Phi(\bullet)$	Gerador progressivo
$\Delta(\bullet)$	Gerador regressivo
$\Xi(\bullet)$	Quantidade de subconjutos S^i visitados
$P(\bullet)$	Probabilidade de ocorrer retrocesso em um S
$T(\bullet)$	Tempo de execução
β	Valor de poda (*bound*)
β_0	Valor de poda inicial
β_i	Valor de poda obtido por S^i
δ	Coeficiente de flutuação
ϵ	Precisão, acuidade, acurácia
γ	Taxa de inconsistência aceitável
ϑ	Limiar de entropia
τ	Limiar para ganho de informação
ξ	Quantidade de flutuações (retrocessos)
$abs(\bullet)$	Retorna o valor absoluto do parâmetro
$arredonda(\bullet)$	Retorna o valor inteiro mais próximo ao parâmetro

Sumário

Resumo

Seleção de Características é um processo que visa a redução da dimensionalidade de um conjunto de dados com o intuito tanto de aumentar a precisão de aplicações em Reconhecimento de Padrões bem diminuir a quantidade de recursos necessários para tal. Esta monografia é um estudo comparativo entre doze métodos para Seleção de Características obtidos pela combinação de seis algoritmos com dois critérios (Taxa de Inconsistência e classificador C4.5), os algoritmos são: Busca Sequencial Progressiva (SFS); Busca Sequencial Regressiva (SFS); Busca Sequencial Flutuante Progressiva (SFFS); Ramificar e Podar (BAB); Ramificar e Podar com com condição de poda alterada (BAB'); e Busca em Profundidade (DEP). A comparação foi realizada de acordo com testes empíricos em dezoito bases de dados, como critério de avaliação dos métodos adotou-se o tempo de execução do método pela precisão obtida pelo classificador, ou seja uma relação de custo-benefício. Por fim, uma discussão sobre a avaliação experimental é apresentada.

Palavras-chave: Seleção de Características, Redução da Dimensionalidade, Classificação, Algoritmos de Busca.

Capítulo 1

Introdução

Com a multiplicação da quantidade de informação gerada e armazenada por sistemas computacionais, processos surgiram para manipular essa informação de modo a transformá-la em conhecimento. O processo de Descoberta de Conhecimento é um deles. Segundo JENSEN & SHEN em [22], Descoberta de Conhecimento é um processo não trivial de identificar padrões válidos, novos, potencialmente úteis e compreensíveis nos dados. Este processo consome um volume de processamento proporcional à quantidade de dados em análise. Por isso, faz-se necessário que o conjunto de dados seja reduzido ao máximo, sem perda de informações relevantes.

Existem duas abordagens para reduzir a dimensionalidade de um conjunto de dados: Extração e Seleção de Características. Segundo DE CAMPOS em [9], Extração de Características é o processo que cria novas características a partir de transformações ou combinações do conjunto de características inicial. Seleção de Características, de acordo com LIU & MOTODA em [31], é o processo que escolhe o subconjunto ótimo de características de acordo com certo critério. Portanto, dado um conjunto de dados com M amostras, formadas por N atributos (características). A aplicação de um método para Seleção de Características sobre o conjunto anterior, resultará em subconjunto com as M amostras, constituídas de $n \leqslant N$ características, sendo que as selecionadas são aquelas que melhor satisfaçam algum critério. Esse trabalho está focado na segunda abordagem.

1.1 Justificativas

Trabalhar com um conjunto de dados menor é benéfico por muitas razões:

- Menos recursos computacionais são necessários;
- Melhoria da qualidade dos dados, pois, dados irrelevantes são ignorados;
- Aumenta a compreensibilidade do conjunto de dados.

1.2 Objetivos

O objetivo desse trabalho é estudar o processo para Seleção de Características aplicado a bases de dados do repositório UCI, ASUNCION & NEWMAN em [4]. Este objetivo se desdobra em:

- Estudo e implementação de métodos para Seleção de Características;
- Comparação entre esses métodos.

1.3 Estrutura do Texto

Essa monografia segue assim estruturada:

- O Capítulo 2 apresenta o processo para Seleção de Características como um problema de busca, que é discutido como uma forma de amenizar o Problema da Dimensionalidade;
- O Capítulo 3 aborda uma série de métodos para Seleção de Características, se aprofundando nos cinco algoritmos que foram implementados. Este capítulo também aborda os dois critérios que foram implementados e conjugados aos algoritmos, sendo que um destes critérios (C4.5) é um classificador e será utilizado para comprovação dos testes comparativos;
- O Capítulo 4 define a metodologia e as bases de dados para a realização dos testes. Ao final desse capítulo são apresentados os resultados dos experimentos realizados;
- Por fim, o Capítulo 5 contém as conclusões e trabalhos futuros a essa pesquisa.

Capítulo 2

Seleção de Características

Este capítulo tem por objetivo contextualizar o leitor sobre o Processo de Seleção de Características (SC). Apresenta definições e conceitos necessários para o entendimento do trabalho.

A tarefa de classificação, realizada por um classificador, é definida como o processo que dado um vetor de características (atributos), associa a tal vetor um rótulo (classe ou categoria) que representa os padrões identificados, [10].

Segundo LIU & MOTODA em [31], SC é o processo que escolhe o subconjunto ótimo de características de acordo com certo critério. Portanto, dado um conjunto de dados com M tuplas, formadas por N características, a aplicação de um método para SC sobre o conjunto anterior resultará em subconjunto com as M tuplas, constituídas de $n \leqslant N$ características, sendo que as características selecionadas são aquelas que melhor satisfaçam algum critério (avaliador).

O Processo de SC tem por objetivo a Redução da Dimensionalidade (RD) de um conjunto de dados. Conforme MARTINS JR. em [34] a RD é necessária para evitar o Problema da Dimensionalidade (PD) ou Comportamento da Curva U, ilustrado na Figura 2.1, adaptada de DE CAMPOS em [9].

De acordo com JAIN, DUIN & MAO em [21], o PD ocorre quando o número de amostras de treinamento para que um classificador obtenha um bom desempenho é uma função exponencial sobre a dimensão dos padrões (o número de características). Portanto, sendo o número de instâncias (M) constante, deve-se determinar a quantidade ideal de características (representado na Figura 2.1 por a) para que um classificador obtenha a acurácia máxima.

Desta forma, o que pode ser observado na Figura 2.1 é que:

- inicialmente ($n < a$), a precisão de um classificador é diretamente proporcional à dimen-

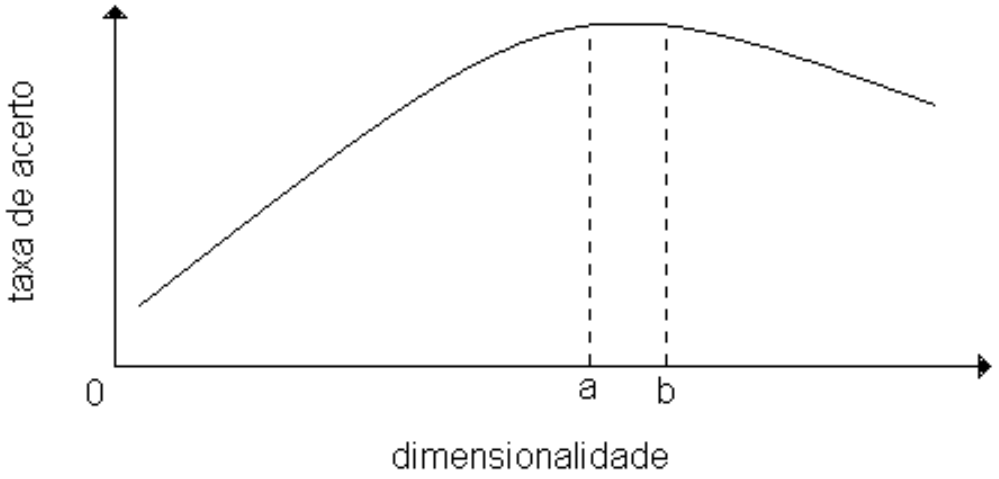


Figura 2.1: Problema da Dimensionalidade

sionalidade (assumindo que o acréscimo de características ocorra em uma ordem tal que características relevantes tenham preferência). Isto ocorre porque enquanto $n < a$, as características não são suficientes para separar as classes;

- em um segundo momento ($a \leqslant n \leqslant b$), o aumento da dimensionalidade não afeta a precisão do classificador, porém, características irrelevantes ou redundantes ao problema são também processadas (desperdício de recursos);

- em um terceiro ($n > b$), a adição de características prejudica a classificação. É neste ponto que o PD ocorre efetivamente, onde a quantidade amostral (M) é insuficiente com relação à quantidade de características fazendo com que a taxa de acerto (precisão) seja reduzida. Esse comportamento é denominado fenômeno do pico (*peaking phenomena*).

Assim, o ideal é que a quantidade de características utilizada seja escolhida de tal maneira que a precisão seja maximizada ($n = a$). Porém, a RD também se faz necessária por outros motivos, segundo LIU & MOTODA em [31]:

- nos casos em que o conjunto de dados for muito grande, recursos computacionais tornam-se insuficientes, não sendo possível para um computador armazenar todos os dados de uma única vez, ou, para que algum algoritmo os processe em tempo hábil (suficiente). Nestes casos, a RD melhora o desempenho do sistema;

- com relação à qualidade dos dados, algumas vezes o conjunto de dados utilizado não foi obtido especificamente para o processo de classificação, ou é oriundo de múltiplas fontes, por isso é possível que características irrelevantes ou redundantes estejam presentes. No processo de SC as características removidas são, prioritariamente, as irrelevantes e as redundantes;

- para facilitar a compreensão sobre os dados, um conjunto de dados com menos atributos é mais simples. Ao reduzir a dimensionalidade dos dados o conjunto de dados torna-se mais conciso e visualizável. Adicionalmente, as regras obtidas na classificação são também mais simples. De uma forma geral os resultados obtidos por qualquer computação posterior ao processo para SC que necessite de supervisão externa (de um especialista) tornam-se mais compreensivos.

2.1 Um Problema de Busca

SC pode ser tratado como um Problema de Busca (LEE em [28] e [29]). A Figura 2.2, adaptada de BLUM & LANGLEY em [5], exemplifica um Espaço de Busca (EB) com quatro características. Cada retângulo representa um estado (possibilidade, configuração, subconjunto). Em cada estado, círculos sólidos representam características selecionadas (círculos não preenchidos representam características não selecionadas). As linhas indicam a transição de um estado a outro, pode-se notar que dois estados conectados diferem exatamente por uma característica. O EB contém dois conjuntos que são os extremos (iniciais), à esquerda existe o conjunto vazio (sem características), e à direita o conjunto completo (contendo todas as características). Subconjuntos de uma mesma camada (dispostos em colunas na Figura 2.2) tem a mesma quantidade de características, cada camada contém exatamente $C_N^n = \frac{N!}{n!(N-n)!}$ subconjuntos totalizando $C_N^0 + C_N^1 + \cdots + C_N^N = 2^N$ estados.

Segundo LIU & MOTODA em [31], diferentes métodos são combinações de três fatores: direção, estratégia e critério; existindo ainda variações com relação ao número de soluções (única ou múltipla); e, também, com relação ao critério de parada (quantidade de características pré-determinada ou limiar da avaliação).

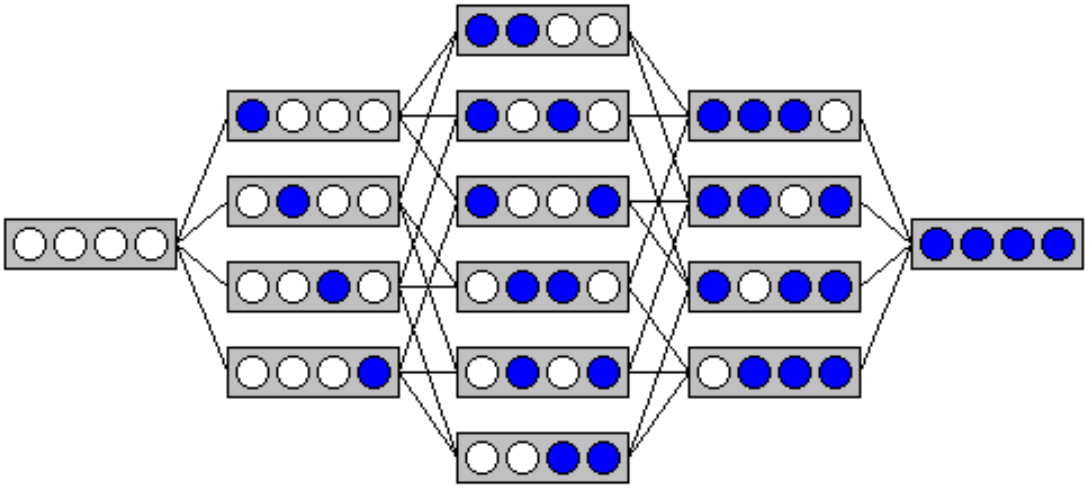


Figura 2.2: Espaço de Busca para $N = 4$

2.1.1 Direção da Busca

Qualquer EB contém dois estados (conjuntos) que são extremos: o conjunto completo, composto por todas as características, e o conjunto vazio, nenhuma característica selecionada. Ambos os conjuntos são importantes, pois determinam os conjuntos iniciais dos algoritmos estudados, conforme a direção utilizada. Conforme LIU & MOTODA em [31] as possíveis direções de busca são:

Progressiva (ou para frente) A busca começa pelo conjunto vazio ($S_{\oslash}$) e a cada iteração o total de características selecionadas no subconjunto do passo atual (S^i) é maior ou igual à quantidade selecionada no subconjunto do passo anterior (S^j), características são escolhidas e unidas ao subconjunto anterior, ou seja, ($|S^i| \geqslant |S^j|$);

Regressiva (ou para trás) Inicia com o conjunto completo (S_*) e a cada iteração o total de características selecionadas pelo passo atual (S^i) é menor ou igual à quantidade selecionada no subconjunto do passo anterior (S^j), ou seja, ($|S^i| \leqslant |S^j|$);

Aleatória Explora o espaço de busca sem uma ordem definida, ou seja, não existe uma lógica determinística que estabeleça a forma com que o espaço é explorado. Exemplos de algoritmos que utilizam esta direção são os derivados da computação genética e métodos

probabilísticos[1];

Bidirecional Explora o espaço de busca em ambas as direções (iniciando tanto pelo conjunto vazio quanto pelo conjunto completo). Pode ser obtida por dois processos executando em paralelo e com direções opostas, cada qual, executando uma instância com metade do espaço de busca, de forma que cada estado seja processado por apenas uma das instâncias. A direção bidirecional não é utilizada por nenhum método implementado durante este trabalho. Pode ser obtida, a título de exemplo, pela conjunção dos métodos SFS e SBS (que são abordados pelas Subseções 3.1.1 e 3.1.2 respectivamente).

Neste trabalho, apenas métodos que utilizam as direções sequenciais (progressivas e regressivas) foram implementados. Uma direção sequencial é preferível a outra nos casos em que a solução estiver mais próxima a um dos extremos, os conjuntos completo e vazio. Neste caso, a escolha da direção implicará em menos iterações da versão do algoritmo que inicie mais próximo da solução. Enquanto a versão progressiva do algoritmo é indicada para a obtenção de um subconjunto contendo poucas características em relação ao conjunto completo ($n \ll \frac{N}{2}$), a versão regressiva do algoritmo é indicado quando se deseja descartar poucas características ($n \gg \frac{N}{2}$). Nos casos em que a solução esteja equidistante dos extremos, ou nesta vizinhança ($n \approx \frac{N}{2}$), se aconselha a adoção da versão progressiva do algoritmo, pois, tende a ser mais custoso avaliar um subconjunto com muitas características, devido a existência de mais dados (características) a processar, conforme atestam JAIN & ZONGKER em [20] e ZONGKER & JAIN em [63].

2.1.2 Estratégia da Busca

O espaço da busca é exponencial com relação à dimensionalidade (2^N). Portanto, avaliar todos os estados pode se tornar impraticável. Deste modo, estratégias surgem para controlar a forma com que o espaço será explorado.

As possíveis estratégias da busca, conforme LIU & MOTODA em [31], são:

Exaustiva ou Força Bruta Quando todo o espaço é explorado. Exemplos são os algoritmos

[1]Métodos que utilizam a direção aleatória não estão presentes neste trabalho, FERRI *et al.* em [12] abordam métodos para SC que utilizam computação genética.

Busca em Profundidade (DEP) e Busca em Amplitude (BRD), o primeiro faz parte do leque de algoritmos estudados (abordados) por este trabalho (ver Subseção 3.1.5);

Completa Quando é garantida a obtenção da solução ótima, portanto, as estratégias exaustivas são também ditas completas. Contudo, existem métodos que encontram a solução ótima sem avaliar todo o espaço da busca, portanto são ditos completos, porém não exaustivos. Ver algoritmo Ramificar e Podar (BAB) na Subseção 3.1.4;

Heurística Explora apenas os caminhos mais promissores (expansão ocorre apenas pelos nodos que obtiverem as melhores avaliações), deteriorando a qualidade da solução em troca de desempenho. A meta de algoritmos que utilizam heurística é obter a melhor solução possível com o mínimo custo computacional. Os algoritmos SFS, SBS e SFFS (nas Subseções 3.1.1, 3.1.2 e 3.1.3, respectivamente) são exemplos abordados para esta estratégia de busca;

Não-determinísticas Exploram o EB de forma aleatória[2]. Portanto, obteem soluções diferentes a cada execução. Oferecem a possibilidade de abortar a execução da busca, retornando a(s) melhor(es) solução(ões) encontrada(s). Abordagens não-determinísticas são indicadas quando a dimensionalidade do conjunto de dados for excessivamente grande, FERRI, KADIRKAMANATHAN & KITTLER em [11].

Apesar da força bruta ser a única forma de garantir a otimalidade da solução, para qualquer caso, maior desempenho pode ser obtido ao se ignorar estados menos promissores da busca, aqueles que contiverem características não discriminantes.

2.1.3 Critério da Busca

Um subconjunto de características é ótimo conforme certo critério. Esta sentença expressa a importância da função critério para o processo de SC, pois é a função critério quem efetivamente qualifica (ou desqualifica) os subconjuntos de características (nodos, S). O sucesso do

[2]Nenhum método presente neste trabalho se vale desta estratégia. Segundo SANTORO em [45], é difícil determinar bons parâmetros de execução para métodos aleatórios, e consequentemente para métodos que utilizem estratégias não-determinísticas. Devido a esta dificuldade em estabelecer os parâmetros aliada à natureza não-determinística, a efetivação dos testes empíricos necessitaria de muitas execuções o que comprometeria o tempo disponível, pois dois dos algoritmos implementados realizam busca completa sendo que se pretende obter a otimalidade por um deles para o efeito de controle na comparação.

processo para SC está fortemente atrelado à avaliação, pois, mesmo que se utilize uma estratégia exaustiva (que teoricamente retorna a solução ótima, a despeito do alto tempo de execução), caso a avaliação utilizada não se assemelhe (seja condizente) ao classificador, a classificação pode ser desastrosa do ponto de vista da precisão, pelo fato do subconjunto ótimo para o critério não ser o melhor subconjunto para a classificação. Este efeito pode ser ainda pior no caso de se adotar estratégias sub-ótimas porque ao invés de seguir os caminhos mais promissores (para o classificador) pode-se percorrer rotas desfavoráveis.

Conforme LIU & MOTODA em [31], critérios podem ser agrupados a partir da métrica utilizada. São elas:

Medidas de Precisão Calculam a precisão preditiva, ou seja, quanto um classificador melhorou sua precisão (ϵ). Uma possibilidade, e a mais natural, é utilizar o próprio classificador como critério, neste caso, diz-se que o método utiliza o modelo *Wraper* de SC ao invés de filtro (ver Seção 2.2);

Medidas de Consistência Procuram o subconjunto mínimo de características que mantenha a máxima consistência com relação ao conjunto completo. Uma inconsistência é definida como duas tuplas com os mesmos valores característicos designadas a classes diferentes. Tanto características irrelevantes quanto redundantes são identificadas;

Medidas de Informação Calculam o ganho de informação ao selecionar uma característica. O ganho de informação é a diferença entre a incerteza do conjunto original (subconjunto gerador, pai ou ainda em expansão) e a incerteza esperada pela adição (ou remoção) de uma característica;

Medidas de Distância (separabilidade, divergência ou discriminantes) Calculam as distâncias entre as classes ao selecionar uma característica, ou seja, quanto ficarão distantes as classes após a seleção da característica;

Medidas de Dependência (associação ou correlação) Calculam o quanto uma característica é relacionada ao valor da classe. Se elas forem estatisticamente independentes, a remoção da característica não alterará a separabilidade entre as classes. Se a cada valor da característica estiver associado a uma classe, a dependência será máxima, e a característica ajudará a determinar a classe.

Em LIU & MOTODA em [31] é afirmado que as três últimas medidas têm a desvantagem de não identificarem características redundantes. Contudo, LEE em [29] e DASH & LIU em [8] afirmam que medidas de dependência também podem ser utilizadas para calcular a redundância entre duas classes, neste caso, calcula-se a dependência entre características, ao invés de características com a classe. Esta contradição entre as literaturas é aparente, pois na primeira referência está escrito que a dependência calcula a correlação (associação) entre duas características, portanto, deixa a entender que se pode reconhecer redundâncias. Porém, deve-se salientar que isto não ocorre de forma natural (pela simples aplicação da medida à classe) como ocorre nas medidas de consistência por exemplo. Para se identificar redundância, por meio de medidas de dependência, é necessário que se calcule a dependência entre as características duas a duas o que deteriora a complexidade do método por um fator de $C_N^2 = \frac{N!}{2!(N-2)!} = \frac{N(N-1)(N-2)!}{2(N-2)!} = \frac{N^2-N}{2} \in O(N^2)$, além de exigir uma implementação específica.

Uma questão importante para o escopo deste trabalho é a monotonicidade. Monotonicidade é uma propriedade presente em algumas funções critério. Ela permite que alguns métodos para SC, em especial o algoritmo BAB (Subseção 3.1.4), obtenham a solução ótima de forma não exaustiva. Segundo NARENDRA & FUKUNAGA em [38], um critério é monotônico quando sua avaliação para um subconjunto de características não é melhor que para um conjunto que o contenha ($S^i \subset S^j, \Psi(S^i) \leqslant \Psi(S^j)$)[3]. Em outras palavras e considerando a direção estritamente regressiva (sequencial para trás), onde características removidas não retornam ao conjunto, a avaliação de um nodo qualquer sempre será melhor ou igual que a avaliação para os seus sucessores. Por exemplo, supondo que a quantidade de características seja igual a quatro ($N = 4$), de acordo com a propriedade monotônica as seguintes afirmações são verdadeiras: $\Psi(S_*) \geqslant \Psi(\{x_1, x_2, x_3\}) \geqslant \Psi(\{x_1, x_2\}) \geqslant \Psi(\{x_1\}) \geqslant \Psi(S_\oslash)$. Considerando o caso em que a quarta característica (x_4) seja irrelevante e que a terceira (x_3) seja redundante, então $\{x_1, x_2\}$ é ótimo. Pode-se representar a afirmação anterior como: $\Psi(S_*) = \Psi(\{x_1, x_2, x_3\}) \geqslant \Psi(\{x_1, x_2\}) > \Psi(\{x_1\}) > \Psi(S_\oslash)$. A remoção da quarta característica (x_4) não afeta a avaliação pois ela é irrelevante, ao se descartar a terceira (x_3) a avaliação será comprometida apenas para métricas que não identifiquem redundância, nos outros casos

[3]Uma avaliação $\Psi(S)$ é melhor quanto maior o seu valor.

(remoção de x_2 e x_1) o valor avaliativo decresce porque estas características são importantes para o escopo do exemplo.

2.2 Modelos para Seleção de Características

Conforme SANTORO em [45], existem dois modelos principais para SC: Filtro e *Wrapper*, existindo ainda o tipo Integrado que é menos abordado. O modelo Integrado ocorre quando o processo para SC é embutido ao classificador, LEE em [28].

SIEDLECKI & SKLANSKY em [49] e KOHAVI em [24] sugerem que quando o objetivo é maximizar a precisão de um classificador medidas de precisão são mais indicadas. E neste sentido, a precisão preditiva pode ser avaliada pelo próprio classificador, o que caracteriza o modelo *Wrapper*. Por este motivo, medidas de precisão estão associadas a este modelo.

Outro modelo também popular é o do tipo Filtro. Em modelos para SC do tipo Filtro, o classificador atua apenas sobre os padrões determinados pelas características selecionados. Por isso, modelos do tipo Filtro consomem menos recursos que os do tipo *Wrapper*, pois, inexiste o classificador que demandaria treinamento e posterior avaliação da precisão obtida a cada subconjunto candidato (gerado), o que reforça a afirmação de que é muito custoso a utilização de um classificador como métrica.

Este trabalho utiliza os dois principais modelos para SC (Filtro e *Wrapper*). Seletores de características como filtros são essenciais por sua ampla aplicação prática (são eficientes). *Wrappers* são importantes para o escopo da comparação por sua natureza eficaz (são acurados, precisos) e servem como uma base sólida para apoiar o resultados obtidos. O terceiro modelo (Integrado) não é estudado em detalhes. KOHAVI & JOHN em [25] citam um exemplo em que uma Árvore de Decisão foi induzida e as características presentes em seus nós de decisão foram tratadas como um subconjunto de características selecionadas que posteriormente foram aplicadas a outro classificador. Porém, existe a possibilidade de características boas a um algoritmo não o serem para outro.

Capítulo 3

Métodos Abordados

Este capítulo tem por objetivo oferecer detalhes mais específicos sobre os métodos estudados neste trabalho.

Na Seção 2.1 o problema de Seleção de Características (SC) foi apresentado como um problema de busca, onde a combinação de três principais aspectos, a direção, a estratégia e o critério, determinam um método em específico. Portanto, existe um número considerável de diferentes métodos para SC. A título de exemplo pode-se citar, além dos métodos presentes no decorrer deste capítulo, os seguintes:

BRD Busca em Amplitude (Largura) (*Breadth-first Search*). É um método exaustivo que explora o espaço de busca camada a camada, no caso da versão progressiva inicia pelo conjunto vazio avaliando-o, em seguida avalia todos os subconjuntos com apenas uma característica, seguido por todos com duas, com três, e assim por diante até que atinja o conjunto completo. Uma fonte que aborda este algoritmo é LIU & MOTODA em [31];

Focus É um método completo baseado no BRD atuando progressivamente. Sua eficácia é decorrente da aplicação de uma medida de consistência como critério (Subseção 2.1.3), o que permite a obtenção da otimalidade. Se diferencia do BRD por interromper a busca ao encontrar um subconjunto consistente (que conterá o mínimo de características)[1];

Relief Apresentado por KIRA & RENDELL em [23]. É um método que atribui pesos (relevância estatística) às características. Originalmente foi desenvolvido para problemas com classes binárias, KONONENKO em [26] melhorou o método para lidar com problemas

[1] As referências ALMUALLIM & DIETTERICH em [1], [2] e [3] são algumas literaturas dos idealizadores sobre o método e algumas variantes (Focus, Focus-1 e Focus-2).

multiclasses, algo já previsto pelos idealizadores do método na sua apresentação. Ao contrário de parte dos algoritmos para SC, Relief não gera subconjuntos de características de forma explícita e opera sobre amostras aleatórias do conjunto de dados. Para cada tupla de dados amostrada durante a execução, encontra-se dois padrões que sejam vizinhos, parecidos, a amostra (e que pertençam a classes diferentes, uma acerta e outra erra a classe da amostra). A lógica empregada é que características relevantes são aquelas que distinguem padrões parecidos. Portanto, uma característica é relevante se tiver valores iguais entre a amostra e a que acerta (a distância é mínima) e diferentes entra a amostra e a que erra (a distância é máxima). Um vetor de distâncias é mantido e contém o montante, para cada característica, sobre as diferenças entre todas as amostras e suas vizinhas (a que acerta e a que erra). Típicamente utiliza-se a distância Euclidiana. Ao final do processo se seleciona as características que ultrapassem um limiar de relevância;

PTA Mais l - Menos r (*Plus* l - *Take Away* r). É um método heurístico que a cada iteração adiciona l características com o algoritmo SFS (Subseção 3.1.1) e remove r características com o algoritmo SBS (Subseção 3.1.2), com os valores l e r passados como parâmetros, sugerido por STEARNS em [56] *apud* PUDIL *et al.* em [39]. Caso $l > r$ a busca se inicia pelo conjunto vazio (ele é progressivo), caso $l < r$ a busca começa pelo completo (é regressivo), sendo que o caso $l = r$ é ignorado (não faz sentido). Com o intuito de remover a passagem dos dois parâmetros foram criados os algoritmos SFFS (Subseção 3.1.3) e SBFS (*Sequential Backward Floating Search*) que na prática calculam os parâmetros l e r dinamicamente. Outro ponto fraco do PTA melhorado pelo SFFS segundo PUDIL *et al.* em [39] é que, ao contrário do SFFS, o PTA realiza retrocesso de forma fixa, nunca superando o valor de l (regressivo) ou r (progressivo). Existe ainda a versão generalizada, GPTA (*Generalized PTA*, PTA Generalizado) que utiliza os algoritmos GSFS e GSBS, adicionando e removendo subconjuntos de características a cada iteração, ao invés de SFS e SBS[2];

LV Corresponde a uma família de métodos probabilísticos, o primeiro da série foi o LVF, Filtro Las Vegas (*LV Filter*). LVF é um método que utiliza uma métrica de consistência (Subseção 2.1.3). O algoritmo LVF recebe como entrada, além do conjunto de dados, uma taxa

[2]Para maiores informações consultar THEODORIDIS & KOUTROUMBAS em [57].

de inconsistência aceitável (γ) e um número de tentativas. No algoritmo inicialmente o conjunto completo é tido como o melhor ($S_{\oplus} \leftarrow S_{*}$). A cada tentativa gera-se um subconjunto aleatório de características (S^{i}), caso a quantidade de características do subconjunto gerado for menor que a do melhor até então e sua avaliação for consistente o suficiente ($|S^{i}| < |S_{\oplus}| \wedge \Psi(S^{i}) < \gamma$), ele é armazenado como a melhor solução $S_{\oplus} \leftarrow S^{i}$. Caso $|S^{i}| = |S_{\oplus}| \wedge \Psi(S^{i}) < \gamma$, S^{i} é também uma solução plausível. LVF retorna como saída uma lista de subconjuntos válidos, todos com a mesma quantidade de características. LVF é apresentado por LIU & SETIONO em [30] existindo ainda a versão LVW (*LV Wrapper*) onde o critério utilizado é um classificador (ao invés de comparar a avaliação à taxa de incosistência se compara com a avaliação do melhor subconjunto). Outra variação é o LVI (*LV Incremental*), uma versão incremental do LVF desenvolvida para grandes bases de dados, o LVI opera apenas com uma fração dos dados por iteração, apresentado por LIU & SETIONO em [32];

OS Busca Oscilatória (*Oscillating Search*), é um algoritmo que melhora uma solução prévia (um subconjunto de características) obtida por outro algoritmo, proposto por SOMOL & PUDIL em [53]. Funciona em ondas progressivas e regressivas que removem e adicionam características de forma estacionária, ao final de cada onda o melhor subconjunto sempre contém a mesma quantidade de características inicial ($|S^{\oplus}| = n$). Partindo de um subconjunto, com n características, remove-se (adiciona-se) o, inicialmente igual à unidade, características com um algoritmo qualquer (pares SFS, SBS; SFFS, SFBS; GSFS, GSBS; ou o PTA), caso o subconjunto com $n \pm o$ características for o melhor subconjunto obtido ($\Psi(S^{n \pm o}) > \Psi(S_{\oplus}^{n \pm o})$), então aplica-se o algoritmo oposto para a obtenção de outro S^{n} e testa-se se houve melhora, em caso positivo faz-se $o \leftarrow 1$ e aplica-se a onda de direção oposta, em caso negativo continua-se como se o teste para $S^{n \pm o}$ houvesse falhado. Caso $\Psi(S^{n \pm o}) \leqslant \Psi(S_{\oplus}^{n \pm o})$, restaura-se S^{n} e repete-se o processo na direção oposta. Sempre que um mesmo S^{n} sofrer tentativas de ondas progressivas e regressivas e falhar em ambas, incrementa-se o e testa-se a condição de parada, o maior que limiar.

Na prática, a aplicação de SC pode exigir que uma série de diferentes métodos sejam executados. Além do mais, os limites de recursos computacionais (geralmente escassos) em relação à quantidade de dados, tende a inviabilizar parte dos métodos para SC. Por consequencia, a

aplicação de SC pode estar atrelada ao problema em questão, ou seja, nem sempre é possível determinar qual método obtém os melhores resultados para um determinado problema.

3.1 Algoritmos

Para a aplicação de SC ser bem sucedida é necessário que ela possa ser aplicada a diferentes contextos. A escolha dos algoritmos abordados neste trabalho foi feita com esse propósito.

Cinco algoritmos estão descritos nas subseções subsequentes, ordenados crescentemente conforme tempos de execuções estimadas segundo as bibliografias SANTORO em [45], LIU & MOTODA em [31] e DE CAMPOS em [9], consequentemente, com relação ao tamanho de instância de dados passível de processamento: Busca Sequencial para Frente (SFS), Busca Sequencial para Trás (SBS), Busca Sequencial Flutuante para Frente (SFFS), Ramificar e Podar (BAB) e Busca em Profundidade (DEP). Estes foram os algoritmos implementados e testados (comparados) durante o desenvolvimento do trabalho. A descrição de cada um deles está ilustrada por um fluxograma; um algoritmo formal; uma computação exemplificadora; e, o cálculo da complexidade.

Normalmente todos os cinco algoritmos são de solução única. As presentes implementações retornam os melhores subconjuntos encontrados, cada qual com uma quantidade de características. A intensão é reduzir a quantidade de execuções do processo para SC quando da utilização prática. Por exemplo, dado um problema real qualquer verifica-se que a aplicação do processo para SC será necessário e *a priori* estima-se que n características sejam suficientes. Se durante os testes, após a execução do algoritmo para SC, nota-se que o ideal era outra quantidade de características então existe a probabilidade de que esta solução já esteja disponível, o que pode propiciar que uma nova execução seja desnecessária.

3.1.1 Busca Sequencial Progressiva (SFS)

O algoritmo de Busca Sequencial Progressiva, ou para Frente, (SFS) é um método Subida de Encosta (*Hill Climbing*, heurística que explora apenas o melhor caminho de busca sem utilizar retrocesso) que utiliza a direção de busca progressiva, inicia pelo conjunto vazio e adiciona uma característica a cada iteração. Por não retroceder tem a desvantagem de ficar preso a um máximo local, descartando o máximo global (efeito *nesting*), o que compromete a qualidade da solução,

algo compensado pelo pouco consumo de recurso computacional (processamento) com relação a outros métodos.

O SFS está ilustrado pela Figura 3.1 e descrito no Algoritmo 3.1 que utiliza o Algoritmo 3.2 ($S^{i+1}SFS.passo(S^i)$). O critério de parada no Algoritmo 3.1 é a quantidade de características máxima que se deseja obter. Ambos adaptados de MARTINS JR. em [34].

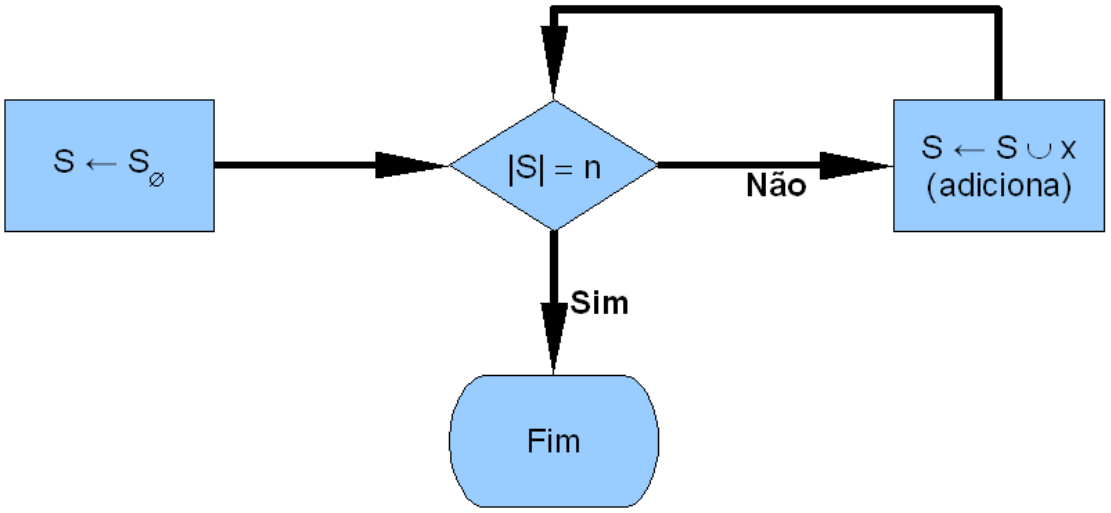


Figura 3.1: Fluxograma do algoritmo SFS

Algoritmo 3.1 Algoritmo SFS

ENTRADA: $0 \leqslant n \leqslant N$
SAÍDA: $S^i_{\oplus}, i = 0, 1, 2, \ldots, n$
1: $S^0_{\oplus} \leftarrow S_{\oslash}$;
2: **PARA** $i \leftarrow 1, 2, \ldots, n$ **FAÇA**
3: $S^i_{\oplus} \leftarrow SFS.passo(S^{i-1}_{\oplus})$;
4: **RETORNE** $S^i_{\oplus}$;

O Algoritmo 3.2 expande um subconjunto (S^i) de forma progressiva (selecionando características) e retorna o melhor subconjunto gerado, caso não seja possível a expansão do subconjunto ($S^i = S_*$) o próprio conjunto gerador é retornado. A notação $\Phi(S^i)$ obtém os subconjuntos gerados pela seleção de uma característica em S^i ($\forall x_j \notin S^i, S^{i+1} = S^i \cup x_j$). Exceto quando S^i for o conjunto completo (S_*), neste caso $\Phi(S_*) = \oslash$.

A execução do SFS expande o conjunto inicial, com nenhuma característica selecionada. Portanto, a expansão do conjunto inicial criará N nodos com apenas uma característica selecionada, ou $N - 1$ características não selecionadas. Após a avaliação dos nodos gerados, o

Algoritmo 3.2 Algoritmo passo SFS

ENTRADA: S^i
SAÍDA: $S_{\oplus}^{j=i+1} \vee S^i$
1: $op \leftarrow \Phi(S^i)$;
2: **SE** $op = \oslash$ **ENTÃO**
3: **RETORNE** S^i;
4: **RETORNE** $S_{\oplus}^{j} \mid \forall op, \Psi(S_{\oplus}^{j}) \geqslant \Psi(op)$;

melhor nodo é escolhido para prosseguir o processo (expansão), porém, a cada nova expansão um nodo a menos será gerado. Esse processo é repetido até que o critério de parada seja satisfeito, subconjunto com n características seja obtido.

A Figura 3.2 mostra uma computação SFS utilizando como critério o classificador C4.5 (Subseção 3.2.2) sobre o EB da base de dados Iris (Subseção 4.2.1 na Página 54) com $N = 4$ e $n = 4$ características. Na Figura 3.2 os subconjuntos estão representados ou como retangulos ou como elipses, o valor abaixo de cada subconjunto é a avaliação. Elipses são os melhores subconjuntos para cada camada, com certa quantidade de características, são as soluções obtidas. As linhas tracejadas e as linhas direcionais indicam a expansão de um subconjunto (geração progressiva de subconjuntos), sendo que as direcionais mostram as transições entre subconjuntos, o melhor subconjunto de cada expansão é escolhido para a próxima iteração.

Sendo N a quantidade de características e n o número de características selecionadas pelo algoritmo. O algoritmo SFS expande n nodos (expansão se refere ao ato de gerar subconjuntos filhos), o nodo inicial (sem características) sempre é expandido. Ao se expandir um nodo é necessário que o critério seja avaliado uma quantidade de vezes igual a quantidade de características não selecionadas, pois este é o número de nodos gerados. A Equação 3.1 é a relação de recorrência que exprime o comportamento do algoritmo SFS

$$T(n) = \begin{cases} 1 & \text{se } n = 0 \\ T(n-1) + (N-n+1) & \text{se } 0 < n \leqslant N \end{cases} \tag{3.1}$$

Aplicando o Método do Desdobramento.

$$\begin{aligned} T(0) &= (1) \\ T(1) &= (1) + (N-1+1) = (1) + (N) \\ T(2) &= (1) + (N) + (N-2+1) = (1) + (N) + (N-1) \end{aligned}$$

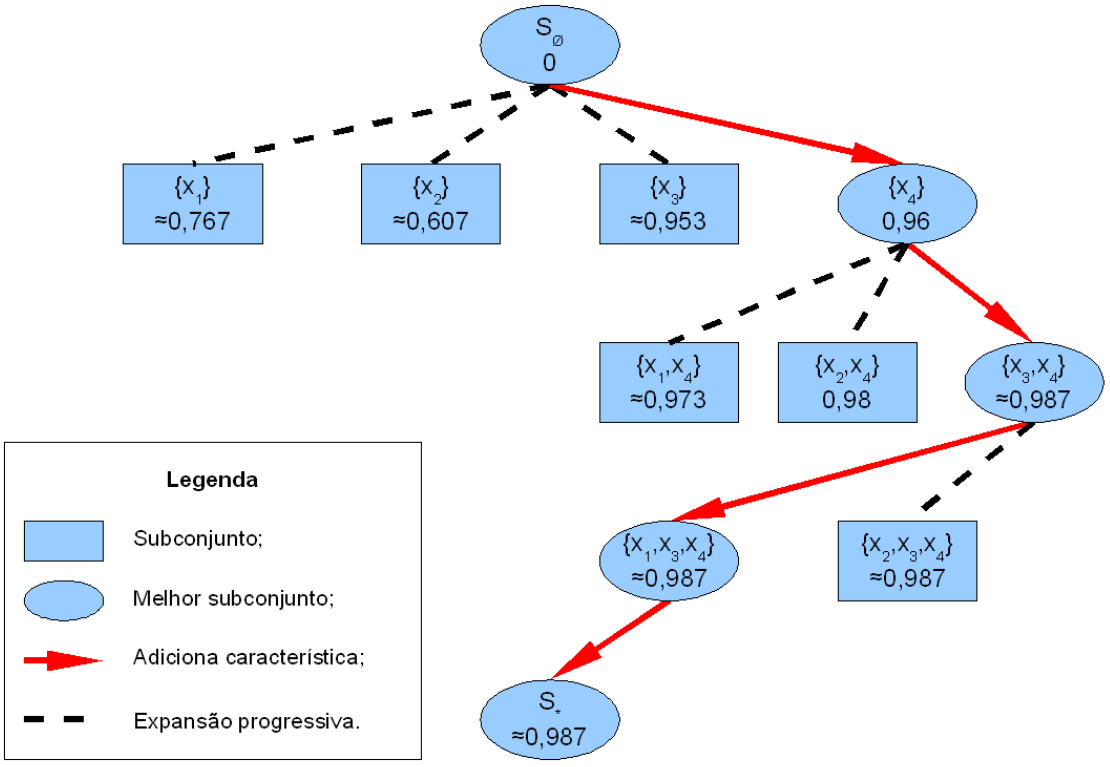


Figura 3.2: Computação SFS (C4.5) para a base Iris ($N = 4$), com $n = 4$

$$
\begin{aligned}
&\vdots \\
T(n) &= (1) + (N) + (N-1) + \cdots + (N-n+1) \\
\\
T(n) &= 1 + \sum_{i=1}^{n} (N - i + 1) \\
&= 1 + n \cdot \frac{(N-n+1) + (N-1+1)}{2} \\
T(n) &= 1 + n \cdot \frac{2 \cdot N - n + 1}{2} \\
T(n) &= \frac{2 \cdot N \cdot n - n^2 + n + 2}{2} \in O(N^2)
\end{aligned}
\tag{3.2}
$$

A Equação 3.2 expressa a complexidade do algoritmo SFS em quantidade de avaliações do critério (AC), quantidade de subconjuntos gerados.

Existe a versão generalizada para este algoritmo GSFS (*Generalized Sequential Forward Search*, Busca Sequencial Progressiva Generalizada) que adiciona algumas características a cada iteração.

3.1.2 Busca Sequencial Regressiva (SBS)

Também chamado de Busca Sequencial para Trás. É similar ao anterior (SFS) no que diz respeito à estratégia da busca (Subida de Encosta) e susceptibilidade ao efeito *nesting* já que também não realiza retrocesso. O que diferencia os algoritmos SBS e SFS é a direção da busca, enquanto o anterior (SFS) utiliza a direção progressiva, este se vale da direção contrária (regressiva), portanto a busca se inicia pelo conjunto completo, removendo características ao invés de adicioná-las. Um fator que influencia a escolha de adotar o algoritmo SFS ou o SBS é a quantidade de características que se deseja selecionar, como discutido na Subseção 2.1.1.

O SBS está ilustrado pela Figura 3.3 e descrito no Algoritmo 3.3 que utiliza o Algoritmo 3.4 ($S^{i-1}SBS.passo(S^i)$). O critério de parada no Algoritmo 3.3 é a quantidade de características mínima que se deseja obter. Ambos adaptados de MARTINS JR. em [34].

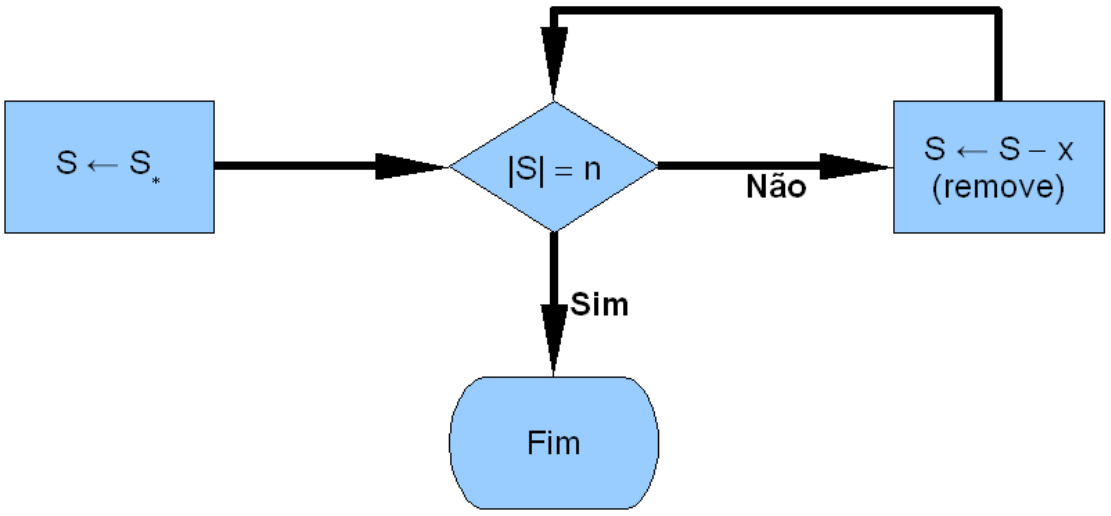


Figura 3.3: Fluxograma do algoritmo SBS

Algoritmo 3.3 Algoritmo SBS

ENTRADA: $0 \leqslant n \leqslant N$
SAÍDA: $S^i_{\oplus}, i = n, n+1, n+2, \ldots, N$
1: $S^N_{\oplus} \leftarrow S_*$;
2: **PARA** $i \leftarrow N-1, N-2, \ldots, n$ **FAÇA**
3: $S^i_{\oplus} \leftarrow SBS.passo(S^{i+1}_{\oplus})$;
4: **RETORNE** $S^i_{\oplus}$;

O Algoritmo 3.4 expande um subconjunto regressivamente (removendo características) e

retorna o melhor subconjunto gerado, caso não seja possível a expansão do subconjunto ($S^i = S_\oslash$) o próprio conjunto gerador é retornado. A notação $\Delta(S^i)$ obtém os subconjuntos gerados pela remoção de uma característica em S^i ($\forall x_j \in S^i, S^{i-1} = S^i - x_j$). Exceto quando S^i for o conjunto vazio ($S_\oslash$), neste caso $\Phi(S_\oslash) = \oslash$.

Algoritmo 3.4 Algoritmo passo SBS

ENTRADA: S^i
SAÍDA: $S_\oplus^{j=i-1} \vee S^i$
1: $op \leftarrow \Delta(S^i)$;
2: **SE** $op = \oslash$ **ENTÃO**
3: **RETORNE** S^i;
4: **RETORNE** $S_\oplus^j \mid \forall op, \Psi(S_\oplus^j) \geqslant \Psi(op)$;

A execução do SBS expande o conjunto inicial, conjunto completo. Portanto, a expansão do conjunto inicial criará N nodos com apenas uma característica não selecionada, ou $N - 1$ características selecionadas. Após a avaliação dos nodos gerados, o melhor nodo é escolhido para prosseguir o processo (expansão), porém, a cada nova expansão um nodo a menos será gerado. Esse processo é repetido até que o critério de parada seja satisfeito, subconjunto com n características seja obtido.

A Figura 3.4 mostra uma computação SBS utilizando como critério o classificador C4.5 (Subseção 3.2.2) sobre o EB da base de dados Iris (Subseção 4.2.1) com $N = 4$ e $n = 0$ características. Na Figura 3.4 os subconjuntos estão representados ou como retângulos ou como elipses, o valor abaixo de cada subconjunto é a avaliação. Elipses são os melhores subconjuntos para cada camada, com certa quantidade de características, são as soluções obtidas. As linhas tracejadas e as linhas direcionais indicam a expansão de um subconjunto (geração regressiva de subconjuntos), sendo que as direcionais mostram as transições entre subconjuntos, o melhor subconjunto de cada expansão é escolhido para a próxima iteração.

Sendo N a quantidade de características e n o número de características selecionadas pelo algoritmo. O algoritmo SBS expande $N - n$ nodos, o nodo inicial (S_*) sempre é expandido. Ao expandir um nodo é necessário que o critério seja avaliado uma quantidade de vezes igual a quantidade de características selecionadas, pois este é o número de nodos gerados. A Equação 3.3 é a relação de recorrência que representa o comportamento do algoritmo SBS.

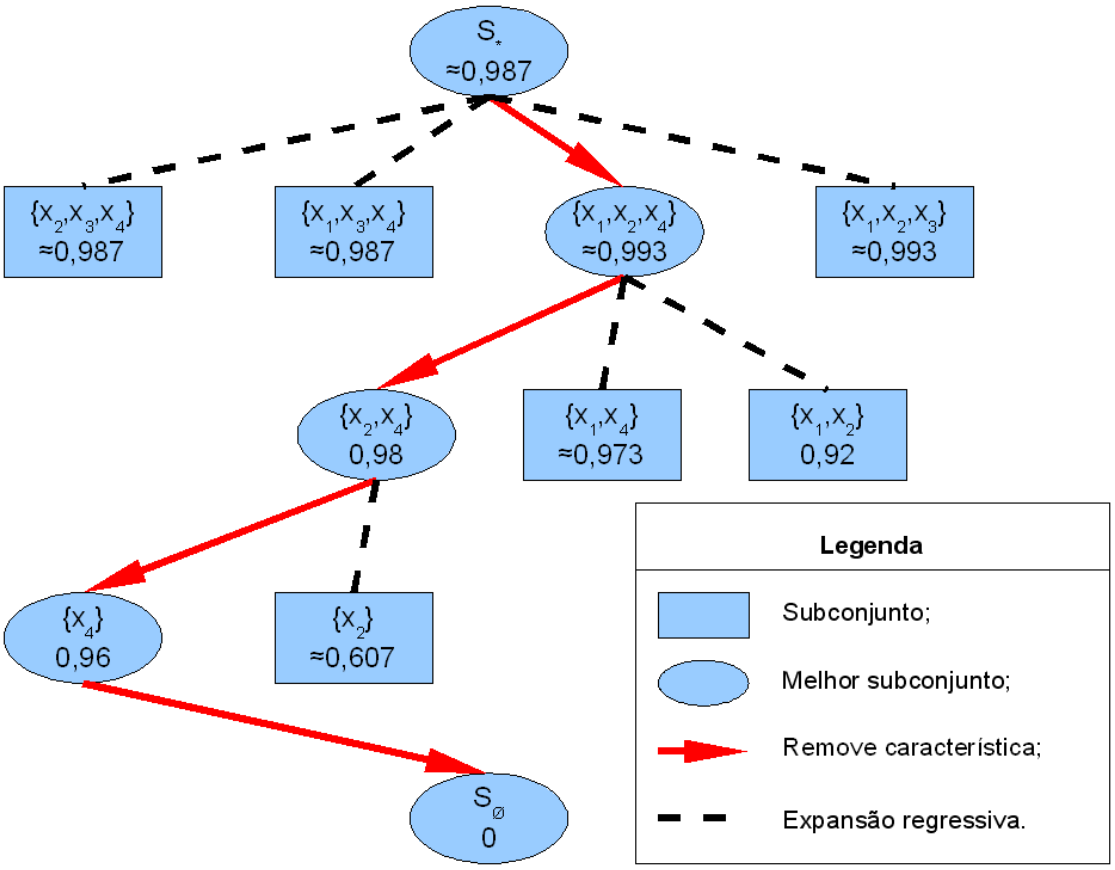


Figura 3.4: Computação SBS (C4.5) para a base Iris ($N = 4$), com $n = 0$

$$T(n) = \begin{cases} 1 & \text{se } n = N \\ T(n+1) + n + 1 & \text{se } 0 \leqslant n < N \end{cases} \tag{3.3}$$

Fazendo $h = N - n$ encontra-se a Equação 3.1.

$$T(h) = \begin{cases} 1 & \text{se } h = 0 \\ T(h-1) + N - h + 1 & \text{se } 0 < h \leqslant N \end{cases}$$

Pela Equação 3.2 na Página 18.

$$T(h) = 1 + h \cdot \frac{2 \cdot N - h + 1}{2}$$

$$\begin{aligned} T(n) &= 1 + (N - n) \cdot \frac{2 \cdot N - (N - n) + 1}{2} \\ &= 1 + (N - n) \cdot \frac{N + n + 1}{2} \end{aligned}$$

$$T(n) \quad = \quad \frac{N^2 + N - n^2 - n + 2}{2} \in O(N^2) \tag{3.4}$$

A Equação 3.4 expressa a complexidade do algoritmo SBS em quantidade de avaliações do critério (AC).

Existe a versão generalizada para este algoritmo GSBS (*Generalized Sequential Backward Search*, Busca Sequencial Regressiva Generalizada) que remove algumas características a cada iteração.

3.1.3 Busca Sequencial Flutuante Progressiva (SFFS)

O algoritmo de Busca Sequencial Flutuante Progressiva, ou para Frente, (SFFS) é uma generalização do algoritmo Mais l - Menos r (PTA) onde os valores l e r são atualizados e determinados dinâmicamente. Definido por PUDIL, NOVOVIČOVÁ & KITTLER em [40] *apud* SOMOL *et al.* em [52]. Neste algoritmo iterações (passos) dos dois algoritmos anteriores (SFS e SBS) são conjugados, de forma que o efeito *nesting* seja amenizado. Segundo ZONGKER & JAIN em [63], esse é o método que melhor combina tempo de execução com qualidade dos resultados devido à capacidade de redução do efeito *nesting*. A versão na direção contrária (regressiva ou para Trás), denomidado de SFBS (*Sequential Floating Backward Search*, Busca Sequencial Flutuante Regressiva) não está implementado neste trabalho.

O SFFS está ilustrado no fluxograma da Figura 3.5 e descrito no Algoritmo 3.5, ambos adaptados de SOMOL *et al.* em [52]. Este algoritmo utiliza o Algoritmo 3.2 da Página 17 (na Linha 4) e o Algoritmo 3.4 da Página 20 (na Linha 12) para aplicar passos únicos dos algoritmos SFS e SBS respectivamente (selecionando e removendo características). O critério de parada é satisfeito quando se obtém a quantidade limiar de características acrescido de δ (Linha 8), o parâmetro δ (coeficiente de flutuação) impede que o SFFS pare ao encontrar o primeiro subconjunto com n características, desta forma, a flutuação também ocorre nesta vizinhança. O coeficiente para a flutuação é um valor inteiro positivo e geralmente é um valor pequeno ($\delta \leqslant 3$).

A execução do SFFS inicia pelo conjunto vazio expandindo-o progressivamente em N subconjuntos com uma característica (todos os subconjuntos possíveis, $C_N^1 = N$) o melhor é armazenado em $S_\oplus^1$. Após um passo do SBS é aplicado em $S_\oplus^1$ obtendo-se $S_\oplus^0 = S_\oslash$ que é descartado. $S_\oplus^1$ é então expandido utilizando o passo SFS que gera $N - 1$ subconjuntos com duas características dentre as quais uma é superior e armazenada em $S_\oplus^2$. Neste ponto já é possível notar a

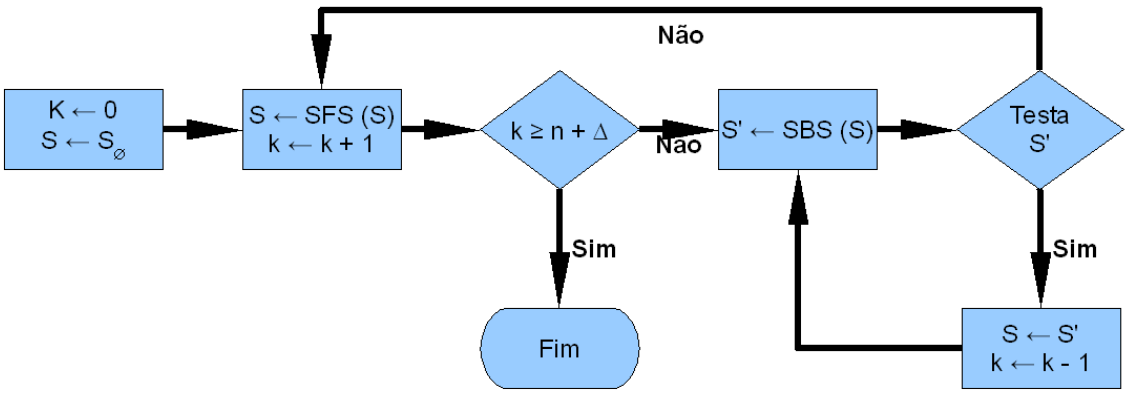


Figura 3.5: Fluxograma do algoritmo SFFS

semelhança existente entre os algoritmos SFS e SFFS (os melhores subconjunto obtidos pelo SFFS até então são os mesmos que seriam obtidos pelo SFS) que caracteriza a direção progressiva do SFFS. Este processamento (aplicação dos passos SFS e SBS) continua até o momento em que o retrocesso seja necessário (enquanto a condição da Linha 13 do Algoritmo 3.5 for satisfeita) então o SFFS se torna semelhante ao SBS, percorrendo o EB de forma regressiva, e continua assim até que o retrocesso não seja mais necessário, então a direção progressiva é retomada, porém, por outra rota do EB que, por via de regra, é mais promissora.

Ao contrário dos algoritmos SFS e SBS, o critério de parada do SFFS é acrescido por um valor de tolerância (δ), permitindo que a flutuação ocorra também em subconjuntos (S) de forma que $|S| \geqslant n$, o retrocesso pode ocorrer enquanto $|S| < n + \delta \wedge \exists S^j$ ainda não visitado, razão pela qual subconjuntos (S) com $|S| \leqslant 2$ nunca serão retrocedidos, conjunto inicial ($S_\oslash$) é único e a expansão de $S_\oslash$ esgota todas as possibilidades com $|S| = 1$. Toda vez que o retrocesso ocorre a probabilidade de novos retrocessos ocorrerem cai, pois, se se supor que $\Xi(S^j) \leqslant C_N^j$ representa a quantia de subconjuntos com j características que já foram avaliados (visitados ou gerados). A probabilidade da ocorrência de um retrocesso em um subconjunto com $i = j + 1$ características é dada por:

$$P(i) = 1 - \frac{\Xi(S^j)}{C_N^j} = 1 - \frac{\Xi(S^j)}{\frac{N!}{j!\cdot(N-j)!}} = 1 - \Xi(S^j) \cdot \frac{j! \cdot (N-j)!}{N!} \tag{3.5}$$

A Figura 3.6 mostra uma computação SFFS utilizando como critério o classificador C4.5 (Subseção 3.2.2) sobre o EB da base de dados Iris (Subseção 4.2.1) com $N = 4$ e $n = 4$ características, sendo que $\delta = 1$. Na Figura 3.6 os subconjuntos estão representados ou como

Algoritmo 3.5 Algoritmo SFFS

ENTRADA: $0 \leqslant n \leqslant N$
ENTRADA: $\delta \geqslant 0$
SAÍDA: $S^i_{\oplus}, i = 0, 1, 2, \ldots, n + \delta$

1: $i \leftarrow 0$;
2: $S^0_{\oplus} \leftarrow atual \leftarrow S_{\oslash}$;
3: **LOOP**
4: $atual \leftarrow SFS.passo(atual)$;
5: $i \leftarrow i + 1$;
6: **SE** $\Psi(atual) > \Psi(S^i_{\oplus})$ **ENTÃO**
7: $S^i_{\oplus} \leftarrow atual$;
8: **SE** $i \geqslant n + \delta$ **ENTÃO**
9: **RETORNE** $S^i_{\oplus}$;
10: $teste \leftarrow$ **FALSO**;
11: **REPITA**
12: $tenta \leftarrow SBS.passo(atual)$;
13: **SE** $\Psi(tenta) > \Psi(S^i_{\oplus})$ **ENTÃO** { $\xi \leftarrow \xi + 1$ }
14: $i \leftarrow i - 1$;
15: $S^i_{\oplus} \leftarrow atual \leftarrow tenta$;
16: **SENÃO**
17: $teste \leftarrow$ **VERDADEIRO**;
18: **ATÉ** $teste$

retangulos ou como elipses, o valor abaixo de cada subconjunto é a avaliação. Elipses são os melhores subconjuntos para cada quantidade de características, são as soluções obtidas. As linhas, independentemente da forma, indicam a expansão de um subconjunto, avaliação de subconjuntos filhos. Linhas retilíneas significam expansões progressivas e linhas curvilíneas (vermelhas) significam expansões regressivas, exceto pela autoconectada em $\{S_*\}$ que é uma expansão progressiva. As linhas direcionais representam transições entre subconjuntos e as linhas adirecionais representam subconjuntos que foram apenas avaliados. As linhas tracejadas direcionais ilustram os retrocessos. A expansão regressiva do subconjunto S_* ocorreu duas vezes nesta computação, na primeira oportunidade ocorreu o retrocesso e na segunda não houve retrocesso porque já haviam se esgotado as possibilidades, conforme pode ser comprovado pela Equação 3.6, que é uma aplicação da Equação 3.5.

(3.6)

$$P(N) \quad = \quad 1 - \Xi(S^{N-1}) \cdot \frac{(N-1)\cdot!(N-N+1)!}{N!} \tag{3.7}$$

$$= \quad 1 - 4 \cdot \frac{(4-1)! \cdot 1!}{4!} = 1 - \frac{4 \cdot 3!}{4!} = 1 - \frac{4!}{4!} = 1 - 1 \tag{3.8}$$

$$= \quad 0 \tag{3.9}$$

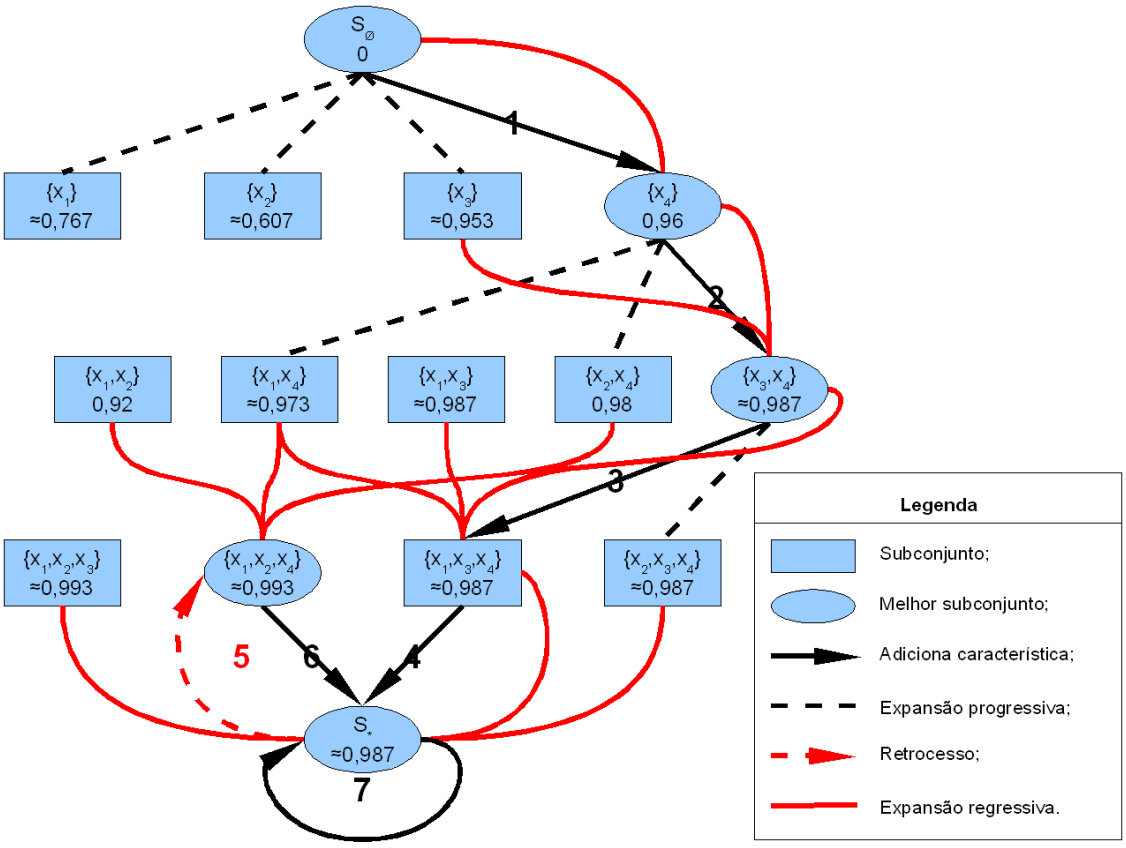


Figura 3.6: Computação SFFS (C4.5) para a base Iris ($N = 4$), com $n = 4$ e $\delta = 3$

Desde que considerada apenas a computação natural (com comportamento idêntico ao SFS), ou seja, sem os retrocessos. O algoritmo SFFS expande $n + \delta$ (conjunto inicial mais $n + \delta - 1$.) subconjuntos progressivamente para satisfazer o critério de parada. E, realiza $n + \delta - 1$ expansões regressivas. A cada retrocesso ocorrem duas expansões regressivas e uma progressiva (uma expansão regressiva por conta do retrocesso em si, mais uma expansão progressiva e outra regressiva para cada avanço em direção ao nível onde o retrocesso começou). Sendo ξ a quantidade de flutuações (retrocessos) em uma execução SFFS, a quantidade de expansões progressivas (EP) e regressivas (ER) totais são dadas pelas Equações 3.10 e 3.11.

$$EP \quad = \quad n + \delta + \xi \tag{3.10}$$

$$ER = n + \delta - 1 + 2\xi \tag{3.11}$$

Em ambas expansões o número de avaliações do critério é igual à quantidade de subconjuntos gerados. Na expansão progressiva este valor é igual ao número de características não selecionadas, não pertencentes ao subconjunto (S) em expansão ($h = N - |S|$). Na regressiva igual à quantidade de características pertencentes ao subconjunto gerador ($|S|$). Quando se aplica as duas expansões a um mesmo subconjunto, N novos subconjuntos são gerados (e avaliados), $N - |S| + |S| = N$. Exceto pelo subconjunto inicial ($S_{\oslash}$) que já gera N subconjuntos de forma progressiva, sempre que um subconjunto é expandido de forma progressiva ele também o é de forma regressiva, exceto pelo conjunto vazio ($S^{\oslash}$). Pode-se unir cada expansão progressiva (Φ) com a sua correspondente regressiva (Δ).

$$\begin{aligned}
EP &= n + \delta + \xi \\
&= 1 + (n + \delta - 1 + \xi) \\
&= (n + \delta - 1 + \xi) \cdot \Phi + N \\
ER &= n + \delta - 1 + 2 \cdot \xi \\
&= (n + \delta - 1 + \xi) + \xi \\
&= (n + \delta - 1 + \xi) \cdot \Delta + \xi \cdot \Delta \\
EP + ER &= (n + \delta - 1 + \xi) \cdot \Phi + N + (n + \delta - 1 + \xi) \cdot \Delta + \xi \cdot \Delta \\
&= (n + \delta - 1 + \xi) \cdot (\Phi + \Delta) + N + \xi \cdot \Delta \\
&\quad \text{Fazendo } (\Phi + \Delta) = N. \\
&= (n + \delta - 1 + \xi) \cdot N + N + \xi \cdot \Delta \\
EP + ER &= (n + \delta + \xi) \cdot N + \xi \cdot \Delta
\end{aligned}$$

Como cada expansão regressiva (Δ) gera no máximo N subconjuntos, as expansões isoladas (que não foram conjugadas com as progressivas) estão limitadas a $O(N)$. A quantidade de avaliações do critério, contando a avaliação do conjunto inicial $S_{\oslash}$, em função do número de retrocessos para o algoritmo SFFS é representada pela Equação 3.12.

$$AC = 1 + EP + ER = 1 + (n + \delta + \xi) \cdot N + \xi \cdot O(N) \tag{3.12}$$

Uma possível relação de recorrência para o comportamento do algoritmo SFFS detalhada acima é a Equação 3.13.

$$T(n) = \begin{cases} 1 & \text{se } n = 0 \\ T(n-1) + N & \text{se não ocorrer retrocesso} \\ T(n+1) + n & \text{se ocorrer retrocesso} \end{cases} \tag{3.13}$$

Aplicando o Método do Desdobramento.

$$\begin{aligned} T(0) &= 1 \\ T(1) &= 1 + N \\ T(2) &= 1 + N + N = 1 + 2 \cdot N \\ &\vdots \\ T(n) &= 1 + (n + \delta + \xi) \cdot N + n \cdot \xi \end{aligned}$$

Como $0 \leqslant n \leqslant N$.

$$\begin{aligned} T(n) &\leqslant 1 + (N + \delta + \xi) \cdot N + N \cdot \xi \in O[(N + \delta + \xi) \cdot N] \\ &\quad \text{Como } \delta \leqslant 3. \\ &\in O[(N + \xi) \cdot N] = O(N^2 + N \cdot \xi) = O[\max(N^2, N \cdot \xi)] \end{aligned} \tag{3.14}$$

A complexidade do algoritmo SFFS com relação à quantidade de avaliações do critério (AC) é obtida pela Equação 3.14 que depende de um fator quadrático sobre N e da quantidade de retrocessos[3] (ξ).

3.1.4 Ramificar e Podar (BAB)

Ramificar e Podar[4] (BAB) é um algoritmo genérico de otimização. NARENDRA & FUKUNAGA em [38] foram os primeiros a abordar o BAB como um método específico para SC. É um algoritmo derivado da Busca em Profundidade (DEP) atuando regressivamente que retorna a solução ótima, com n características ($S^n_{\otimes}$), caso a função critério aplicada seja monotônica,

[3]Existe a possibilidade de se limitar a quantidade de retrocessos, com certo valor de significancia, pelo uso da Equação 3.5. Durante a execução deste trabalho não foi possível desenvolver tal raciocínio.

[4]Tradução adotada para *Branch and Bound*. SANTOS em [46] sugere que outra tradução possível seja *Ramificar e Limitar*.

o que não impede que o algoritmo também seja aplicado em conjunção com critérios não monotônicos. HAMAMOTO *et al.* em [17] avaliam o algoritmo BAB (sem monotonicidade) em comparação com a busca exaustiva e concluem que o algoritmo BAB obtem êxito (boa taxa de reconhecimento) mesmo sem a garantia da propriedade monotônica. Se diferencia do algoritmo DEP por dois aspectos:

1. A árvore de busca BAB não é exaustiva em seus ramos intermediários, quando $n < i < N$, ver a discussão dobre as subfiguras da Figura 3.9;

2. Considerando a notação $C^n_\oplus$ como o melhor subconjunto com n características obtido até o momento, e que $\beta = \Psi(S^n_\oplus)$. Sempre que um subconjuto gerado $S^i \mid \Psi(S^i) < \beta$ a árvore é podada, subconjuntos que descendem de S^i são ignorados.

O algoritmo BAB está definido no Algoritmo 3.6 que é uma implementação recursiva (na Linha 23) para o fluxograma ilustrado na Figura 3.7, ambos adaptados de NARENDRA & FUKUNAGA em [38]. O Algoritmo 3.6 presume o uso de um vetor ($disp$) com N valores lógicos, tal vetor tem a função de informar quais características podem ser deselecionadas ao se visitar um subconjunto qualquer. As melhores soluções são armazendas em $S^i_\oplus, i = n, n+1, \ldots, N$. Essas variáveis ($disp$ e $S^i_\oplus$) bem como o valor de poda (β) devem ser tratadas como variáveis globais, comuns a toda recursão. Inicialmente, é assumido que:

- todas as características estão habilitadas a remoção $disp[1 \ldots N] \leftarrow V$);
- o subconjunto inicial (S_*) da árvore de busca BAB contém $t = n + 1$ ramos;
- $\beta \leftarrow -\infty$, valores maiores para β fazem com que ramos sejam descartados antes (aumenta a efeciência do algoritmo). Os subconjuntos soluções tem $\Psi(S^i_\oplus) \geqslant \beta$;
- $pai \leftarrow S_*$ (conjunto completo).

A execução do BAB inicia pelo conjunto completo ($pai \leftarrow S_*$), primeiro subconjunto visitado. Sempre que um subconjunto é visitado, caso seja necessário, atualiza-se $S^i_\oplus$ (na Linha 2). Se o subconjunto for uma folha (comparação da Linha 3), atualiza-se β (na Linha 5) conforme necessidade. Quando o subconjunto não for uma folha enumera-se todos os subconjuntos possíveis (no laço da Linha 8), ordenando decrescentemente conforme avaliação (Linha 11). No laço

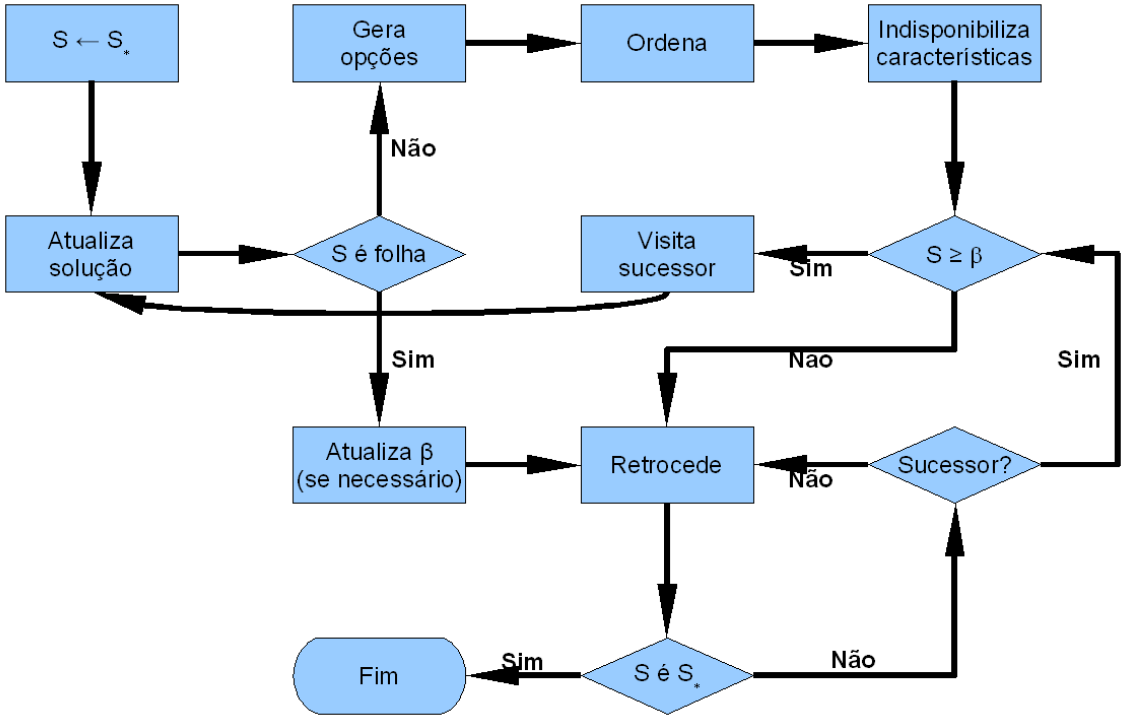


Figura 3.7: Fluxograma do algoritmo BAB

da Linha 12, são desabilitadas as características (removidas) que determinam os t subconjuntos que orientam a(s) ramificação(ões). Após, ocorrem as ramificações (laço da Linha 20), onde os t melhores subconjuntos são candidatos a serem expandidos recursivamente, caso tenham avaliação inferior ao valor de poda ($\Psi(atual) < \beta$) o subconjunto, bem como seus descendentes, são ignorados; caso contrário a comparação da Linha 22 é satisfeita. No processo de ramificação os melhores subconjuntos são visitados de forma prioritária e gerando menos ramos $(1, 2, \ldots, t)$, aumentando a probabilidade do valor de poda (β) crescer rapidamente, pois, ao ramificar subconjuntos mais promissores existe uma tendência em obter subconjuntos melhores nas folhas da árvore.

A Figura 3.8 mostra uma computação BAB utilizando como critério o classificador C4.5 (Subseção 3.2.2) sobre o EB da base de dados Iris (Subseção 4.2.1) com $N = 4$ e $n = 2$ características, sendo que $\beta = -\infty$. Na Figura 3.8 os subconjuntos estão representados ou como retângulos ou como elipses, o valor abaixo de cada subconjunto é a avaliação. Elipses são os melhores subconjuntos para cada camada, com certa quantidade de características, são as soluções obtidas. As linhas, sejam elas tracejadas ou direcionais, indicam a expansão de um

Algoritmo 3.6 Algoritmo BAB

ENTRADA: $0 \leqslant n \leqslant N$
ENTRADA: $1 \leqslant t \leqslant n+1$
ENTRADA: pai, subconjunto visitado
SAÍDA: $S^i_{\oplus}, i = n, n+1, \ldots, N$

1: **SE** $\Psi(pai) > \Psi(S^{|pai|}_{\oplus})$ **ENTÃO**
2: $S^{|pai|}_{\oplus} \leftarrow pai$;
3: **SE** $n = |pai|$ **ENTÃO**
4: **SE** $\Psi(pai) > \beta$ **ENTÃO**
5: $\beta \leftarrow \Psi(pai)$;
6: **SENÃO**
7: $L \leftarrow \oslash$;
8: **PARA** $i \leftarrow 1, 2, \ldots, N$ **FAÇA**
9: **SE** $disp[i]$ **ENTÃO**
10: $L \leftarrow L \cup (pai - x_i)$;
11: $Ordena(L)$; { decrescentemente }
12: **PARA** $i \leftarrow 1, 2, \ldots, t$ **FAÇA**
13: $atual \leftarrow L[i]$;
14: **PARA** $j \leftarrow 1, 2, \ldots, N$ **FAÇA**
15: **SE** $disp[j]$ **ENTÃO**
16: **SE** $x_j \notin atual$ **ENTÃO**
17: $disp[j] \leftarrow$ **FALSO**;
18: $op[i] \leftarrow j$;
19: **QUEBRA**;
20: **PARA** $i \leftarrow 1, 2, \ldots, t$ **FAÇA**
21: $atual \leftarrow L[i]$;
22: **SE** $\Psi(atual) \geqslant \beta$ **ENTÃO** { Poda }
23: $BAB(n, i, atual)$;
24: $disp[op[i]] \leftarrow$ **VERDADEIRO**;
25: **RETORNE** $S^i_{\oplus}$, caso $\Psi(\bullet)$ seja monotônica $S^n_{\oplus} = S^n_{\otimes}$;

subconjunto pela remoção de uma característica conforme o vetor de disponibilidade. As linhas direcionais sólidas (não tracejadas) ilustram a visitação (ramificação) de um subconjunto, os números indicam a ordem de visitação, e as linhas direcionais tracejadas representam as podas. Nota-se nos subconjuntos $\{x_1, x_3\}$ e $\{x_2, x_3\}$ que as atualizações do valor de poda estão explicitadas. O processo recursivo ocorre de cima para baixo e da direita para a esquerda[5] (ordem de visitação). Nota-se na Figura 3.8 que tal árvore enumera todos os subconjuntos possíveis com n características. Caso a propriedade monotônica seja válida ($\Psi(C^j) \leqslant \Psi(C^i) \forall C^j \subset C^i$), a otimalidade para n características estará garantida ($C^n_{\otimes}$ é encontrado), pois, os subconjuntos C^n

[5]Percorrer a árvore BAB da direita para a esquerda é mais eficiente, pois propicia que β seja atualizado antes.

ignorados implicitamente por uma poda em C^i não superam um $C^n_{\oplus}$ previamente encontrado, ou seja, $\Psi(C^{j=n}) < \Psi(C^n_{\oplus}) = \beta \mid \forall C^j$ que descenda de um C^i podado ($\Psi(C^i) < \beta$).

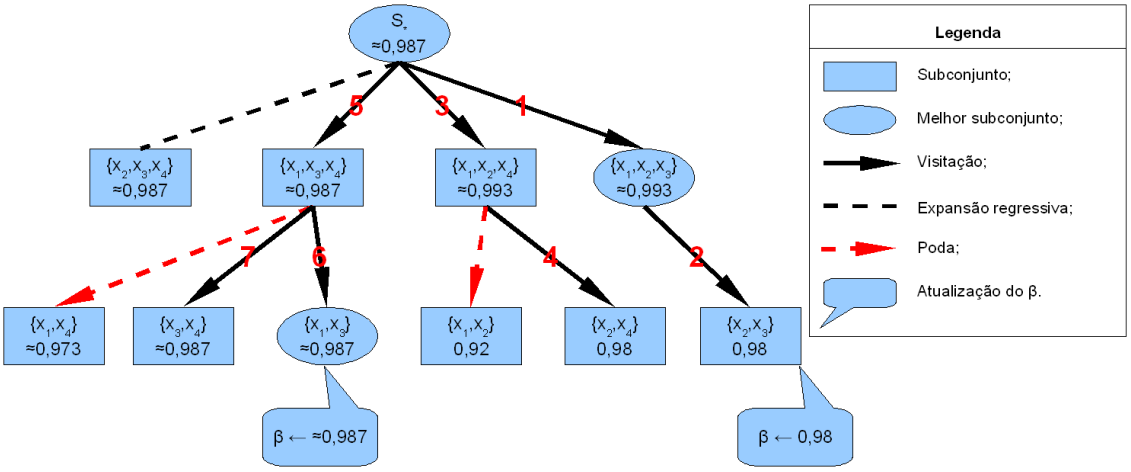


Figura 3.8: Computação BAB (C4.5) para a base Iris ($N = 4$), com $n = 2$

A Figura 3.9 ilustra algumas árvores de busca BAB para $N = 4$ ($n = 0, 1, 2, 3$), respectivamente. Pode-se notar algumas propriedades sobre a árvore de busca BAB, como:

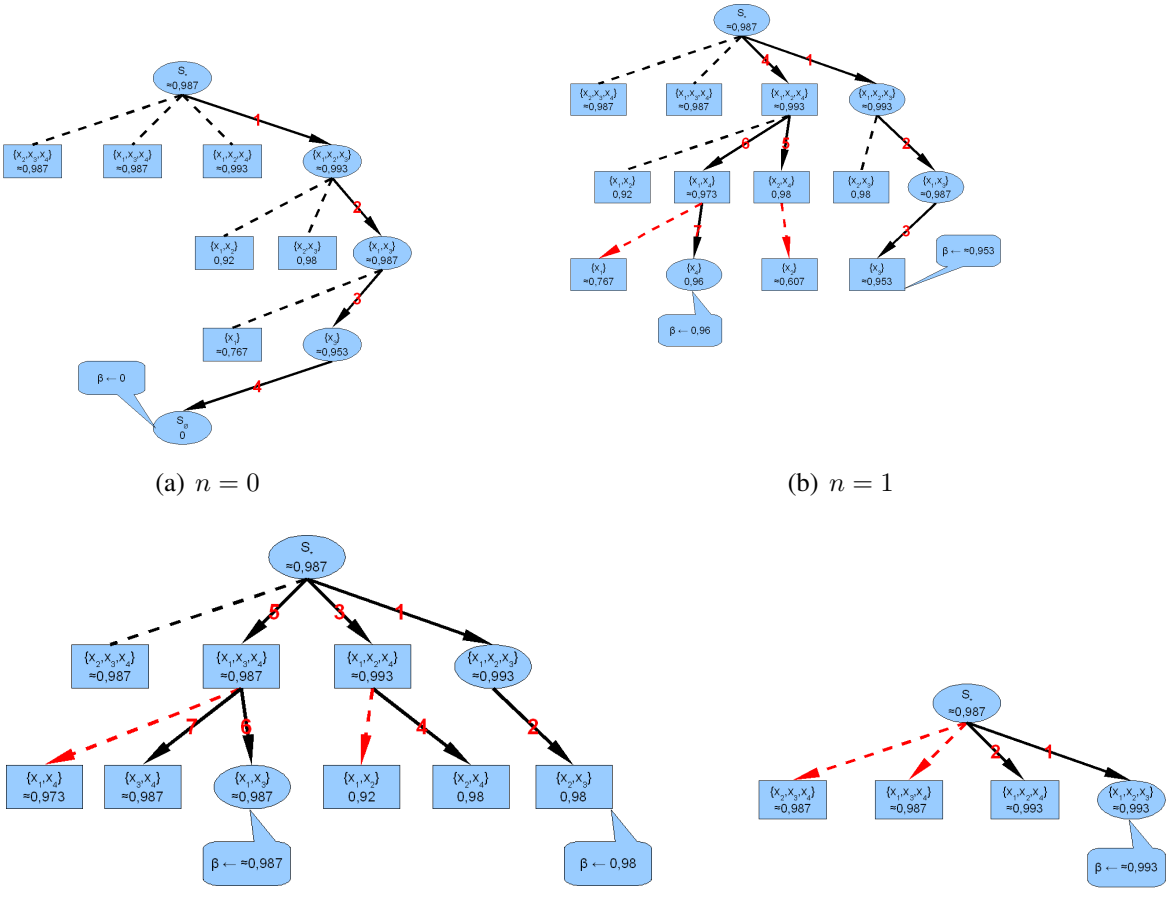
1. não contém subconjuntos repetidos;

2. não é exaustiva em seus ramos internos, por exemplo, na Figura 3.9(a) o subconjunto $\{x_2, x_3, x_4\}$ é gerado (e avaliado) mas não é visitado e o subconjunto $\{x_4\}$ não é processado em momento algum;

3. exceto pelos subconjuntos folhas que não geram descendentes, cada subconjunto visitado S tem $1 \leqslant t \leqslant n+1$ ramos, o t-ésimo ramo (pela ordem de visitação) de um subconjunto tem exatamente t ramos. E, todo S que seja um ramo expande (e avalia) $1 \leqslant j \leqslant N$ subconjuntos, de forma que o último de seus t ramos expanda (e avalie) um subconjunto a menos $(j-1)$, o penúltimo $j-2$ e assim por diante;

4. o conjunto inicial (S_*), desde que $n \neq N$, enumera (avalia) todos os N subconjuntos possíveis com $N-1$ características, sendo que exatamente $t = n+1$ destes subconjuntos são visitados, tornam-se ramos.

Nota-se no Algoritmo 3.6 que o teste para realizar a poda (Linha 22) é da forma $\Psi(atual) \geqslant \beta$, ou seja, poda ocorre quando $\Psi(atual) < \Psi(S^n_{\oplus}) = \beta$, desde que β já tenha sido atualizado.



(a) $n = 0$ (b) $n = 1$

(c) $n = 2$ (d) $n = 3$

Figura 3.9: Árvores de busca BAB (C4.5) para a base Iris ($N = 4$), com $n = 0, 1, 2, 3$

Esse teste faz uso da propriedade monotônica para melhorar a eficiência do algoritmo, mas assumindo a validade da propriedade monotônica, se um subconjunto solução já está determinado e sua avaliação é β continuar com a ramificação nos casos em que $\Psi(atual) = \beta$ não melhora a solução, como pode ser observado na Figura 3.10 em que os subconjuntos $\{x_2, x_4\}$ e $\{x_3, x_4\}$ são podados, na Figura 3.8 que ilustra a mesma computação para o algoritmo BAB original estes subconjuntos são também avaliados. Portanto, é ainda possível maximizar o efeito da propriedade monotônica pela substituição da comparação da Linha 22 por $\Psi(atual) > \beta$, o que evita a ramificação quando $\Psi(atual) = \Psi(S^n_{\oplus})$. Essa simples alteração pode melhorar o desempenho do BAB, porém, pode deteriorar a qualidade da solução quando a monotonicidade não for válida, contudo, a otimalidade em presença da monotonicidade ainda é garantida. Uma versão com a alteração está também implementada e é testada em comparação com o BAB. Essa

versão alterada é simbolizada por BAB'.

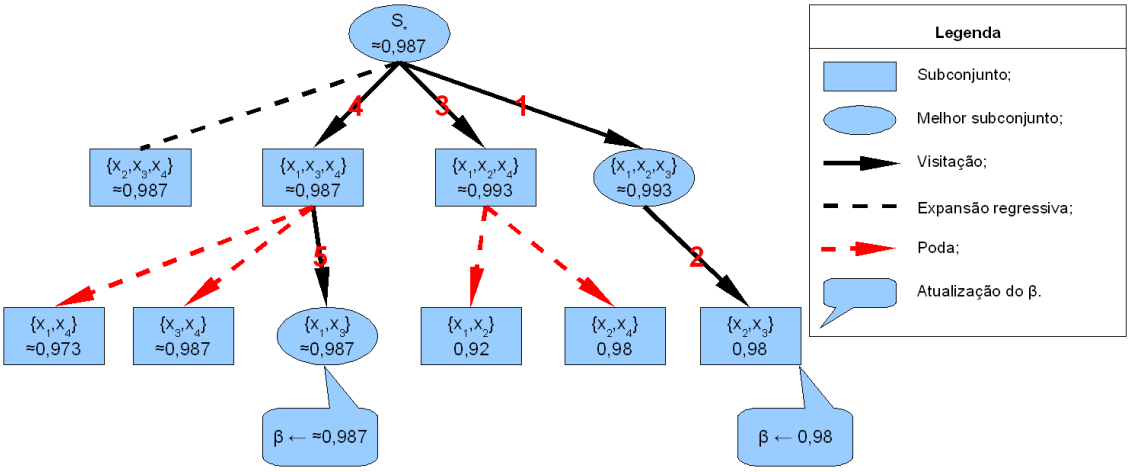


Figura 3.10: Computação BAB' (C4.5) para a base Iris ($N = 4$), com $n = 2$

Existe uma série de variantes para o algoritmo BAB que, pela tradução de RONCATTI [44], são denominadas:

BAB básico Se diferencia do BAB ordenado pela inexistência da ordenação (Linha 11 no Algoritmo 3.6), NARENDRA & FUKUNAGA em [38];

BAB ordenado Representado pelo Algoritmo 3.6, é a versão utilizada neste trabalho. É caracterizado pela ordenação que ocorre a cada subconjunto visitado, de forma que rotas (ramos) mais promissores sejam expandidos de forma prioritária com o intuito de maximizar os efeitos benéficos da poda, reduzir a quantidade de chamadas à função critério, NARENDRA & FUKUNAGA em [38];

BAB sub-ótimo É ainda mais eficiente que o BAB ordenado (Algoritmo 3.6), a diferença reside na forma com que a atualização do valor de poda (β) é realizada, a atualização não ocorre apenas nas folhas, ela ocorre em todos os níveis, pois, existem uma série de $\beta_i \mid 0 \leqslant i \leqslant N$, caso $i \geqslant n, \beta_i = \Psi(S^i_{\oplus})$, caso contrário $\beta_i = \beta = \Psi(S^n_{\oplus})$. A poda ocorre sempre que um subconjunto $S \mid \Psi(S^{i \geqslant n}) < \beta_{i-l}$, onde $l \geqslant 0$ é uma constante inteira denominada fator de *lookahead* e que determina o valor de poda que deve ser considerado. A eficiência é maior quanto menor é o valor de l. Caso $l \geqslant n - 1$ o BAB sub-ótimo torna-se o BAB, descrito pelo Algoritmo 3.6. Se $l = 0$ o BAB sub-ótimo aproxima a computação do algoritmo SBS, NARENDRA & FUKUNAGA em [38];

Árvore de busca mínima Nesta versão do BAB se ignora o cálculo do critério sempre que um subconjunto visitado gerar apenas um ramo ($t = 1$) avançando a busca diretamente para o próximo subconjunto até que se atinja a folha, YU & YUAN em [62] *apud* RONCATTI em [44]. Por exemplo, na computação representada pela Figura 3.8, a avaliação dos subconjuntos $\{x_1, x_2, x_3\}$, $\{x_1, x_2\}$ e $\{x_3, x_4\}$ não seriam calculadas;

BAB rápido É uma versão modificada do BAB ordenado utilizando a árvore de busca mínima e que realiza previsões do critério para reduzir ainda mais a quantidade de chamadas à função critério, SOMOL *et al.* em [54] *apud* RONCATTI em [44]. A maior parte das avaliações reais ocorrem em subconjuntos folhas (para atualizar β) e em prováveis situações de poda;

BAB com previsão parcial É uma modificação do BAB rápido, sem o uso da árvore de busca mínima, que se vale das previsões apenas para a etapa de ordenação (subconjuntos visitados são avaliados pelo critério), SOMOL, PUDIL & GRIM em [55] *apud* RONCATTI em [44].;

Busca da direita para a esquerda A árvore de busca BAB não enumera todas as possibilidades de poda, ou seja, a árvore continua a enumerar subconjuntos $S^j \subset S^i \Rightarrow \Psi S^j \leqslant \Psi(S^i) \mid S^i$ é um subconjunto já podado, portanto S^j também pode ser podado. CHEN em [7] aprimorou a algoritmo BAB para possibilitar que tais ramos também sejam podados. Todo ramo S^i podado é armazenado, sempre que um subconjunto S^j é visitado verifica-se se $S^j \subset S^i$. A busca tem que realizada obrigatoriamente da direita para a esquerda, caso contrário, tal regra ($S^j \subset S^i \mid S^i$ é um subconjunto já podado) nunca será válida;

BAB adaptativo NAKARIYAKUL & CASASENT em [36] propuseram esta versão do BAB com algumas propriadades adicionais: ordenação única das características antes da construção da árvore de acordo com a significância das características, calculada utilizando o algoritmo SFS; obtenção inical de um bom subconjunto (maximização de β) com a aplicação de uma Busca Sequencial Flutuante (SFFS ou SFBS); a busca se inicia em um nível mais elevado (não se inicia por S_*); utiliza uma nova estratégia adaptativa para pular a busca alguns níveis abaixo na árvore de forma que computações desnecessárias

da função critério sejam evitadas, isso ocorre quando um subconjunto S é visitado e $\Psi(S) \gg \beta$, então presume-se que não ocorrerá poda nos descendentes diretos de S ($\not\exists x_j \mid \Psi(S - x_j) < \beta$) e a busca pode continuar nos descendentes indiretos de S (descendentes de $S - x_j$)[6];

Floresta Em RONCATTI em [44] é apresentada uma nova estratégia para o algoritmo Ramificar e Podar que foi integrado à versão adaptativa, mas que também pode ser utilizada um conjunção com outras abordagens do BAB. A estratégia Floresta recebe este nome por utilizar uma série de árvores BAB para realizar a busca. Cada árvore da floresta opera sobre uma porção das características, e durante a construção de uma nova árvore as características menos relevantes são ignoradas e a árvore gerada tem suas dimensões reduzidas. A busca ocorre da última árvore construída para a primeira, o que leva à avaliação dos melhores subconjuntos mais cedo, e aumenta a probabilidade de ocorrem podas em níveis mais altos da árvore original, aumentando assim a eficácia das podas, uma vez que mais subconjuntos são descartados.

3.1.5 Busca em Profundidade (DEP)

É um método exaustivo que nesta monografia foi implementado utilizando a direção progressiva. Por ser exaustivo analisa todos os subconjuntos possíveis, o que torna este algoritmo proibitivo para um número grande de características. Porém, sempre encontra a solução ótima.

O algoritmo Busca em Profundidade (DEP) está ilustrado no fluxograma da Figura 3.11 e descrito no Algoritmo 3.7 implementado de forma recursiva, ambos adaptados de LIU & MOTODA em [31]. A implementação tradicional do algoritmo DEP permite que nodos sejam visitados mais de uma vez. No Algoritmo 3.7 cada subconjuntos é visitado apenas uma vez, isto é feito utilizando a mesma lógica que os autores do Algoritmo BAB (NARENDRA & FUKUNAGA em [38]) usam em seu algoritmo com a utilização do vetor de disponibilidade ($disp$ no Algoritmo 3.6). Ao invés de, para cada subconjunto visitado gerar todos os subconjuntos possíveis pela seleção de uma característica, comportamento implementado pela rotina $SFS.passo()$ no Algoritmo 3.2 na Página 17 o que faria que subconjuntos fossem revisitados (desperdício da recursos), sempre que visita-se um subconjunto seleciona-se apenas as carac-

[6]NAKARIYAKUL & CASASENT em [37] é uma fonte que compara uma série de versões BAB.

terísticas $x_{N-j+1} \cdots x_N$, onde j é uma variável de controle que substitui o uso do vetor de disponibilidade. No Algoritmo 3.7 é assumido que inicialmente $j \leftarrow N$ e $pai \leftarrow S_{\oslash}$ (conjunto vazio).

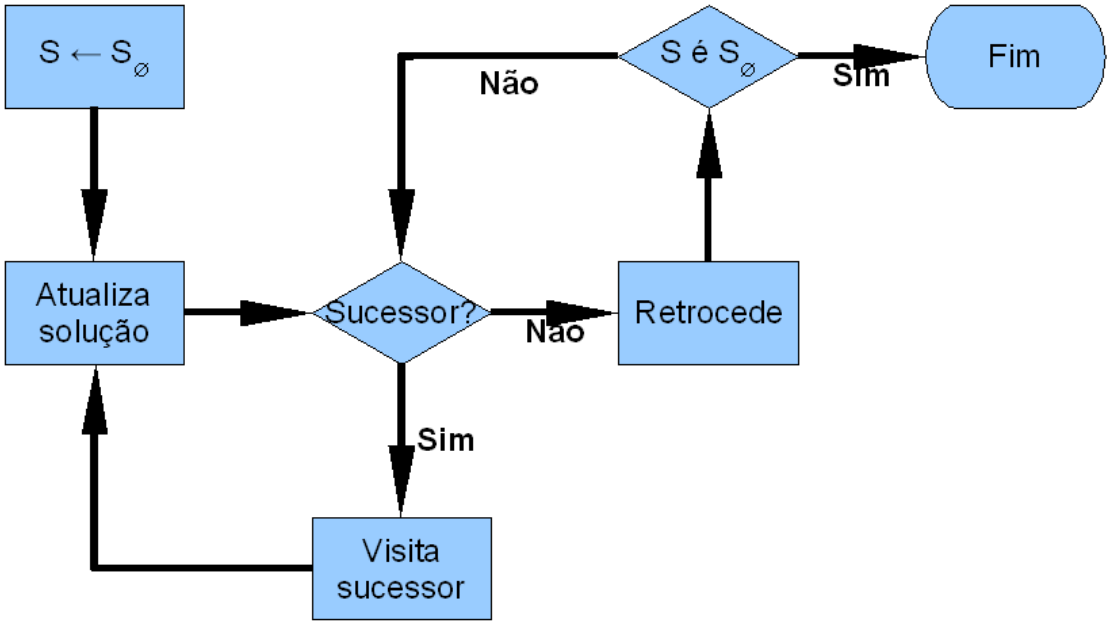


Figura 3.11: Fluxograma do algoritmo DEP

Algoritmo 3.7 Algoritmo DEP

ENTRADA: $1 \leqslant j \leqslant N$
ENTRADA: pai, subconjunto visitado
SAÍDA: $S^n_{\otimes}, n = 0, 1, \ldots, N$

1: **SE** $\Psi(pai) > \Psi(S^{|pai|}_{\otimes})$ **ENTÃO**
2: $\quad S^{|pai|}_{\otimes} \leftarrow pai$;
3: $i \leftarrow N - j + 1$;
4: **ENQUANTO** $i \leqslant N$ **FAÇA**
5: $\quad j \leftarrow j - 1$;
6: $\quad DEP(j, pai \cup x_i)$;
7: $\quad i \leftarrow i + 1$;
8: **RETORNE** $S^n_{\otimes}$;

A execução do DEP se inicia pelo conjunto vazio ($S_{\oslash}$). Sempre que um subconjunto é visitado sua avaliação é comparada com a avaliação do melhor subconjunto obtido até então na mesma camada (com a mesma quantidade de características). Caso seja superior, se atualiza. Após, os subconjuntos descendentes são visitados de forma recursiva, um de cada vez e obede-

cendo a regra supracitada (selecionando características de forma que não se visite subconjuntos repetitivamente).

A Figura 3.12 mostra uma computação DEP utilizando como critério o classificador C4.5 (Subseção 3.2.2) sobre o EB da base de dados Iris (Subseção 4.2.1) com $N = 4$ características. Na Figura 3.12 os subconjuntos estão representados ou como retangulos ou como elipses, o valor abaixo de cada subconjunto é a avaliação. Elipses são os melhores subconjuntos para cada camada, com certa quantidade de características, são as soluções obtidas. As linhas direcionais ilustram a visitação a um novo subconjunto pela seleção de uma característica, os números nas linhas direcionais indicam a ordem em que os nodos são visitados.

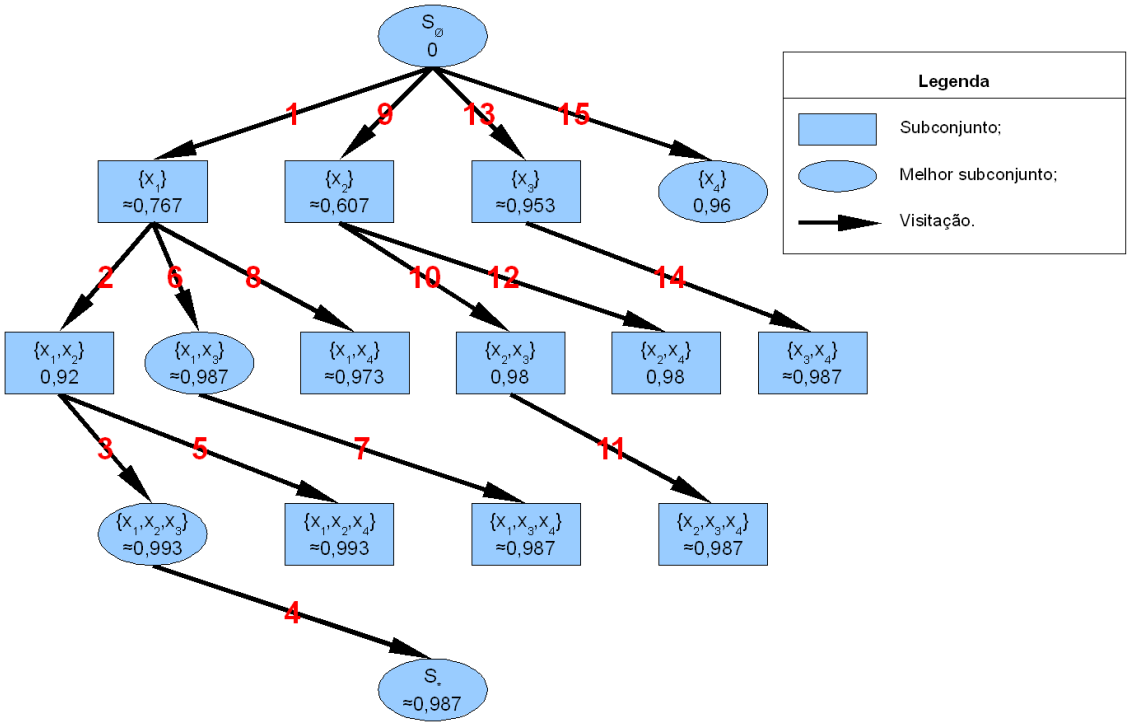


Figura 3.12: Computação DEP (C4.5) para a base Iris ($N = 4$)

O DEP visita (e avalia) todos os subconjuntos exatamente uma vez, como em cada camada $0 \leqslant i \leqslant N$ (formada por todos S^i) existem exatos $C_N^i = \frac{N!}{i!(N-i)!}$ subconjuntos, a quantidade de avaliações do critério (AC) do DEP é dada por:

$$AC = C_N^0 + C_N^1 + \cdots + C_N^N$$

Um Binômio de Newton é dado por:

$$
\begin{aligned}
(x+y)^N &= C_N^0 x^N.y^0 + C_N^1 x^{N-1} y^1 + \cdots + C_N^N x^0 y^N \\
&\quad \text{Fazendo } x = y = 1. \\
(1+1)^N &= C_N^0 + C_N^1 + \cdots + C_N^N = 2^N \\
AC &= 2^N \in \Theta(2^N)
\end{aligned} \tag{3.15}
$$

Pela Equação 3.15 a complexidade do Algoritmo DEP é $\Theta(2^N)$.

3.2 Funções Critério

Nas subseções seguintes são descritas as funções critérios utilizadas neste trabalho (Taxa de Inconsistência e classificador C4.5), sendo que apenas a primeira foi implementada, pois para a utilização da segunda foi reaproveitado o código-fonte que VOLTOLINI em [59] usou em sua monografia, que por sua vez foi derivado da implementação de FRANÇOIS em [14] (jaDTi - Decision Trees: a Java implementation). Cada critério conta com uma breve descrição teórica, representação algorítmica da implementação utilizada e sua complexidade para o pior caso.

A metodologia utilizada para o cálculo da complexidade do critério Taxa de Inconsistência é baseada na abordagem da análise de complexidade pessimista contida em [58], que leva em consideração construções algorítmicas (atribuições, sequências, condicionais, laços). De uma forma simplificada o custo de cada construção algorítmica conforme a metodologia é:

Atribuição Custo para avaliar a expressão mais o custo de transferir o resultado à variável;

Sequência Soma dos custos de cada componente;

Condicional Custo para avaliar a expressão lógica mais o maior custo entre os dois blocos disjuntivos possíveis (**ENTÃO** ou **SENÃO**);

Laços Quantidade de repetições multiplicada pela soma dos custos do bloco e da avaliação da expressão lógica.

A operação básica considerada para o cálculo das complexidades dos critérios é a quantidade de Acessos aos Padrões (AP), ou seja, as complexidades são expressas em função de M.

3.2.1 Taxa de Inconsistência

A Taxa de Inconsistência é uma métrica de Consistência, portanto, identifica tanto características irrelevantes quanto redundantes.

Segundo LIU & MOTODA em [31], a medida de Consistência é monotônica, portanto, pode ser conjugada ao algoritmo BAB para obter a otimalidade, com relação ao critério, com determinada quantidade de características ($S^n_{\otimes}$), porém, tem a desvantagem de apenas operar sobre dados discretos, sendo que para aplicar uma métrica de consistência sobre dados contínuos é necessários que os dados sejam discretizados. O método discretizante aplicado neste estudo foi o método dos Intervalos de Tamanhos Iguais (EWI) apresentado na Subseção 3.3.1.

A Taxa de Inconsistência é calculada pela aplicação das regras:

1. duas instâncias são inconsistentes se contiverem os mesmo valores característicos exceto pela classe. Por exemplo, considerando $N = 4$ e um subconjunto $S^2 = \{x_2, x_3\}$, duas possíveis instâncias inconsistentes são $\{1, 0, 2\}$ e $\{1, 0, 3\}$, onde o terceiro valor corresponde à classe das instâncias que diferem entre si;

2. contagem de inconsistências para cada grupo de instâncias compatíveis (valores característicos identicos) é o total de instâncias compatíveis menos a maior ocorrência da classe. Por exemplo, se i instâncias são compatíveis e todas são mapeadas para duas classes distintas, sendo que cada classe tem respectivamente $j, k \mid i = j + k$ instâncias, então a contagem de inconsitências é igual a $i - \max(j, k)$;

3. a Taxa de Inconsistência é igual à soma de todas as contagens de inconsistências (para cada instância existente) dividida pelo total de instâncias (M).

O Algoritmo 3.8, adaptado de LIU & MOTODA em [31], descreve a implementação utilizada para a Taxa de Incosistência. Este algoritmo retorna uma porcentagem ($0 \leqslant \Psi(S) \leqslant 1$) e, para fins de compatibilidade com os algoritmos para SC, esta avaliação é maior quanto melhor for o subconjunto. Neste algoritmo é assumido que:

- a tabela de espalhamento (Linha 2) comporta todos os padrões e que sua função de espalhamento não mapeia dois padrões diferentes a um mesmo compartimento, e, que cada

compartimento tem acesso a um vetor de valores inteiros que é utilizado para contar as ocorrências dos valores de classe de um mesmo padrão;

- a base de dados (Υ) esta acessível (Linha 5) e que este acesso retorna um padrão (ρ) compatível com o subconjunto avaliado, obtém apenas os valores característicos presentes no subconjunto mais o valor da classe;
- λ denota a quantidade de classes possíveis a um padrão qualquer;
- a notação $\Lambda(\rho)$ extrai o valor de classe (o alvo) do padrão.

Algoritmo 3.8 Algoritmo do Critério Taxa de Inconsistência

ENTRADA: S, subconjunto que será avaliado
SAÍDA: $\Psi(S)$

```
 1: SE S ≠ S_∅ ENTÃO
 2:    TabelaHash Tab[M][λ];
 3:    Tab[1,2,...,M][1,2,...,λ] ← 0;
 4:    PARA i ← 1,2,...,M FAÇA
 5:       ρ ← Υ[i]; { Acessa o i-ésimo padrão conforme S }
 6:       SE ρ ∈ Tab ENTÃO { Busca na Tabela de Espalhamento }
 7:          T[ρ] ← Tab[ρ][Λ(ρ)] + 1;
 8:       SENÃO
 9:          Tab ∪ ρ; { Adiciona padrão à Tabela de Espalhamento }
10:          Tab[ρ][Λ(ρ)] ← 1;
11:    inconsist ← 0;
12:    PARA i ← 1,2,...,M FAÇA
13:       soma ← maior ← 0;
14:       PARA j ← 1,2,...,λ FAÇA
15:          soma ← soma + Tab[i][j];
16:          SE Tab[i][j] > maior ENTÃO
17:             maior ← Tab[i][j];
18:       inconsist ← inconsist + (soma − maior);
19:    RETORNE 1 − inconsist/M;
20: SENÃO
21:    RETORNE 0;
```

A complexidade para se calcular a Taxa de Inconsistência de um subconjunto qualquer, desde que as operações sobre a tabela de espalhamento (Linhas 6 e 9) e leitura de um padrão (Linha 5) garantam complexidade próximas a uma constante, depende principalmente dos dois laços principais (Linha 4 e Linha 12) cada um executando M vezes. A Tabela 3.1 relaciona as linhas do algoritmo com suas complexidades de execução.

Tabela 3.1: Complexidades parciais do Algoritmo 3.8

Linha do Algoritmo 3.8	**Complexidade**
1	$O(1)$
2	$O(1)$
3	$O(M \cdot \lambda)$
4	$O(1)$
5	$O(1)$
6	$O(1)$
7	$O(1)$
9	$O(1)$
10	$O(1)$
11	$O(1)$
12	$O(1)$
13	$O(1)$
14	$O(1)$
15	$O(1)$
16	$O(1)$
17	$O(1)$
18	$O(1)$
19	$O(1)$
21	$O(1)$

Considerando a notação $T(\bullet)$ o custo de execução da construção algorítmica de uma linha, a complexidade no pior caso para o Algoritmo 3.8 é calculada utilizando a abordagem de TOSCANI & VELOSO em [58] por

$$
\begin{aligned}
AP &\in T(1) \\
T(6) &= O(N) + \max[T(7), T(9) + T(10)] \\
&= O(1) + \max[O(1), O(1) + O(1)] = O(1) \\
T(4) &= M \cdot [T(5) + T(6)] \\
&= M \cdot [O(1) + O(1)] = O(M) \\
T(16) &= O(1) + O(1) = O(1) \\
T(14) &= \lambda \cdot [T(15) + T(16)] \\
&= \lambda \cdot [O(1) + O(1)] = O(\lambda) \\
T(12) &= M \cdot [T(13) + T(14) + T(18)]
\end{aligned}
$$

$$
\begin{aligned}
&= M \cdot [O(1) + O(\lambda) + O(1)] = O(M \cdot \lambda) \\
T(1) &= O(1) + \max[T(2) + T(3) + T(4) + T(11) + T(12) + T(19), T(21)] \\
&= O(1) + \max[O(1) + O(M \cdot \lambda) + O(M) + O(1) + O(M \cdot \lambda) + O(1), O(1)] \\
&= O(1) + \max[O(M\lambda), O(1)] = O(M \cdot \lambda) \\
&\quad \text{como } M \gg \lambda. \\
AP &\in O(M)
\end{aligned} \tag{3.16}
$$

A Equação 3.16 denota a complexidade para avaliar um subconjunto qualquer pelo critério Taxa de Inconsistência, que é $O(M)$ Acessos aos Padrões (AP).

3.2.2 Classificador C4.5

É um classificador que constroi uma Árvore de Decisão (AD). O C4.5 foi idealizado por QUINLAN em [42] e realiza o treinamento de forma parecida ao ID3 (*Iterative Dichotomiser 3*) definido por QUINLAN em [41]. A implementação utilizada é derivada do trabalho de VOLTOLINI em [59] que usa a biblioteca jaDTi[7] (*Decision Trees: a Java implementation* desenvolvida por FRANÇOIS em [14]) para inferir um classificador que é testado e determina uma precisão estimada. Neste trabalho, o C4.5 é utilizado com duas finalidades: como critério do processo para SC; como classificador para a obtenção da precisão das soluções obtidas pela aplicação dos métodos para SC sobre as bases de dados testadas no Capítulo 4.

Uma AD é uma estrutura em árvore que tem a função de, dado um vetor de padrões ρ determinar a classe a qual o padrão pertence ($\Lambda(\rho)$). Em uma Árvore de Decisão cada nó é:

- ou uma folha que contém um identificador para uma classe do problema, quando se aplica o processo de classificação sobre um padrão ρ, atingir um nó folha significa que a classificação é conclusiva e que ρ pertence à classe determinada pela folha;
- ou um nó de decisão que indexa uma característica do padrão e aponta para outras subárvores, durante a classificação do padrão ρ o valor característico de ρ é comparado a uma série de possibilidades que indicam o caminho, para a raiz de uma subárvore, que deverá ser seguido para que se conclua a classificação do padrão.

[7] A biblioteca jaDTi é disponibilizada sobre a licença GPL (*GNU General Public License*).

O C4.5 utiliza a abordagem de Divisão e Conquista para construir a AD, QUINLAN em [43]. O uso da abordagem Dividir para Conquistar com o intuito de induzir Árvores de Decisão é derivada do trabalho de HUNT, MARIN & STONE em [19] *apud* QUINLAN em [43]. O processo que constrói a Árvore de Decisão é recursivo e descrito como segue, considerando $\Upsilon^0 = \Upsilon$ o subconjunto de padrões que é passado como parâmetro para o início da recursão (raiz da árvore):

- se Υ^i contém apenas padrões que pertencem a uma mesma classe $c_i \mid 1 \leqslant i \leqslant \lambda$, então o nó é uma folha que identifica a classe dos padrões, c_i;
- se Υ^i não contém nenhum padrão, então o nó é uma folha que identifica o padrão com a maior frequência no nó anterior (Υ^{i-1}), o nó que originou (que aponta para) esta folha;
- se Υ^i é formado por padrões de classes disitintas, então Υ^i é particionado em j subconjuntos de padrões residuais (Υ^{i+1}), que de acordo com o valor de uma determinada característica x_k, cada novo subconjunto de padrões satisfaz alguma condição sobre x_k, são os testes que serão realizados durante o processo de classificação. Novas subárvores são geradas, um para cada subconjunto de padrões, de forma que a avaliação de tais condicionais levem à raiz das subárvores geradas.

Os testes que são gerados nos nós de decisão construídos pelo classificador C4.5 podem ser de dois tipos dependendo do tipo de dado que é armazenado nas características $x_i \mid 1 \leqslant i \leqslant N$:

1. para características discretas, os testes são da forma $x_i = v$ com uma ramificação para cada possibilidade;
2. para características contínuas, os testes assumem a forma $x_i \leqslant v^{lim}$, existem apenas duas possibilidades (ramos onde $x_i \leqslant v^{lim}$ ou $x_i > v^{lim}$). Para encontrar v^{lim} que maximiza o particionamento, o subconjunto de padrões Υ^j é ordenado de acordo com os valores de x_i resultando em uma sequência ordenada de t elementos distintos ($v_1, v_2, \ldots, v_t$, num total de $t-1$ partições), cada par de elementos adjascentes determina um valor limiar em potencial ($v^{lim} = \frac{v_k + v_{k+1}}{2}$) sendo que o escolhido é aquele que maximizar o critério de particionamento sobre alguma característica x_i, todas as características são testadas.

Conforme QUINLAN em [43] e WU *et al.* em [60], o classificador C4.5 pode adotar como critério de particionamento tanto o Ganho de Informação ($Ganho(\bullet, \bullet)$) quanto a Taxa de Ganho ($\frac{Ganho(\bullet,\bullet)}{Part(\bullet,\bullet)}$), o Ganho de Informação é o critério originalmente utilizado pelo classificador ID3. Ambos critérios de particionamento (Ganho e Taxa de Ganho), são medidas baseadas em informação (no caso Entropia, fundamentada por SHANNON em [47][8] *apud* MARTINS JR. em [34]). A seguir, estão definidas as fórmulas utilizadas pelo C4.5 da forma que foi originalmente definida:

Equação 3.17 Proporção de elementos do vetor de alvos que pertencem a determinada classe;

Equação 3.18 Incerteza Residual para o (Entropia do) vetor de classes;

Equação 3.19 Ganho de Informação quando se escolhe o teste em v, considerando que $V^i \subset V \mid \forall c_j \in V, c_j \in V^i$ sse $c_j = \Lambda(\rho) \wedge \rho$ satisfaz o i-ésimo teste;

Equação 3.20 Informação de uma Partição.

$$p(V, c) = \frac{1}{|V|} \times \sum_{i=1}^{|V|} 1 \text{ sse } c_i = c \tag{3.17}$$

$$Info(V) = -\sum_{i=1}^{\lambda} p(V, i) \times \lg p(V, i) \tag{3.18}$$

$$Ganho(V, v) = Info(V) - \sum_{i=1}^{t-1} \frac{|V^i|}{|V|} \times Info(V^i) \tag{3.19}$$

$$Part(V, v) = \sum_{i=1}^{t-1} \frac{|V^i|}{|V|} \times \lg \frac{|V^i|}{|V|} \tag{3.20}$$

O Algoritmo 3.9 descreve a implementação do C4.5 adotado pelo jaDTi. Existem algumas diferenças pontuais da implementação adotada e do algoritmo C4.5 original, as modificaçoes identificadas são detalhadas após a apresentação da rotina. No algoritmo é assumido que:

- os valores das classe são valores naturais que respeitam a regra $1 \leqslant c_i \leqslant \lambda$. Isso deve-se a uma simplificação de notação, na prática a biblioteca jaDTi requer que $0 \leqslant c_i \leqslant (\lambda - 1)$;

[8] HAYKIN em [18] comenta a existência de uma série de fontes que contém o referido artigo, a citar SHANNON & WEAVER em [48], SLEPIAN em [50], SLOANE & WYNER em [51]. Nenhuma desta bibliografia foram encontradas para consulta.

- λ denota a quantidade de classes possíveis a um padrão qualquer.

Algoritmo 3.9 Algoritmo C4.5

ENTRADA: Υ^i, subconjunto de padrões
ENTRADA: ϑ, limiar de entropia
ENTRADA: τ, limiar para ganho de informação
SAÍDA: AD, Árvore de Decisão constrída

1: $entropia \leftarrow Info(\Lambda(\Upsilon^i))$;
2: **SE** $entropia \leqslant \vartheta$ **ENTÃO** { Poda }
3: $AD \leftarrow folha(c_{\max} \mid \forall c_j \in \Lambda(Upsilon^i), f(\Lambda(Upsilon^i), c_{\max}) \geqslant f(\Lambda(Upsilon^i), c_j))$;
4: **SENÃO**
5: $ganho \leftarrow \max Ganho(\Lambda(\Upsilon[][j]), v^{lim})$; { Maximiza o Ganho de Informação }
6: **SE** $ganho \times |\Upsilon^i| \leqslant \tau$ **ENTÃO** { Poda }
7: $AD \leftarrow folha(c_{\max} \mid \forall c_j \in \Lambda(Upsilon^i), f(\Lambda(Upsilon^i), c_{\max}) \geqslant f(\Lambda(Upsilon^i), c_j))$;
8: **SENÃO**
9: $AD \leftarrow ramo(x, v^{lim})$;
10: $AD.maior \leftarrow C4.5(\Upsilon^{i+1}, \vartheta, \tau\S) \mid \Upsilon^{i+1} \subset \Upsilon^i \wedge \forall \rho \in \Upsilon^i, \rho \in \Upsilon^{i+1}$ sse $\rho^x \geqslant v^{lim}$;
11: $AD.menor \leftarrow C4.5(\Upsilon^{i+1}, \vartheta, \tau\S) \mid \Upsilon^{i+1} \subset \Upsilon^i \wedge \forall \rho \in \Upsilon^i, \rho \in \Upsilon^{i+1}$ sse $\rho^x < v^{lim}$;
12: **RETORNE** AD;

O classificador C4.5 da biblioteca jaDTi realiza pré-podas, aborta a construção da AD, conforme os limiares de entropia (ϑ, Linha 2) e de ganho de informação (τ, Linha 6), nos dois casos é criado um nó folha com a classe mais frequente.

Na Linha 5, os ganhos de informação para cada limiar (v^{lim}) para cada característica (x_j) são calculados em uma única passada, as frequências das classes dos padrões são calculadas *on the fly* pelo incremento (decremento) das frequências maiores e menores que v^{lim}.

Algumas peculiaridades desta implementação são:

1. pelas recursões do Algoritmo 3.9, Linhas 10 e 11, nota-se que a implementação utilizada trata todas as características como contínuas, mesmo quando elas forem discretas. Essa abordagem pode gerar inconsistências caso os valores destas características não mantiverem relação de ordem umas com as outras, isso ocorre quando a característica representa um valor simbólico. Nota-se também que o teste realizado para particionar o subconjunto de padrões nas recursões ($<$ ou $\geqslant$) é diferente do proposto originalmente ($\leqslant$ ou $>$);

2. a Entropia[9] é calculada de uma forma diferente da Equação 3.18, utilizado a frequência

[9]Segundo MARTINS JR. em [34] no cálculo da Entropia, por convenção, caso a $p(\bullet, \bullet) = 0, \ln(0) = 0$. Na

($f(V,c)$) ao invés da proporção e $\ln$ no lugar de $\lg$ (Equação 3.21). Ambas são a mesma fórmula, como pode-se demonstrar por:

Considerando:

$$f(V,c) = \sum_{i=1}^{|V|} 1 \text{ sse } c_i = c$$

Partindo da definição original, Equação 3.18.

$$\begin{aligned}
Info(V) &= -\sum_{i=1}^{\lambda} p(V,i) \times \lg p(V,i) \\
&= -\sum_{i=1}^{\lambda} \frac{f(V,i)}{|V|} \times \lg \frac{f(V,i)}{|V|} \\
&= \frac{1}{|V|} \times -\sum_{i=1}^{\lambda} f(V,i) \times [\lg f(V,i) - \lg |V|] \\
&= \frac{1}{|V|} \times -\sum_{i=1}^{\lambda} f(V,i) \times \left[\frac{\ln f(V,i)}{\ln 2} - \frac{\ln |V|}{\ln 2}\right] \\
&= \frac{1}{|V| \times \ln 2} \times \left\{\sum_{i=1}^{\lambda} [f(V,i) \times \ln |V|] - \sum_{i=1}^{\lambda} f(V,i) \times \ln f(V,i)\right\} \\
&= \left[|V| \times \ln |V| - \sum_{i=1}^{\lambda} f(V,i) \times \ln f(V,i)\right] \times \frac{1}{|V| \times \ln 2} \qquad (3.21)
\end{aligned}$$

O Algoritmo 3.10, adaptado de VOLTOLINI em [59], obtém a precisão do classificador C4.5 a partir de dois subconjuntos de padrões, um para treinamento e outro para teste. Quando o C4.5 é utilizado como critério, todo o conjunto de dados é utilizado tanto para treinamento quanto para teste ($\Upsilon^{TR} = \Upsilon^{T} = \Upsilon$), apesar de MARTINS em [33] afirmar que nestes casos a precisão obtida pela classificação é denominada aparente e não representam boas estimativas de precisão, essa questão é retomada na Subseção 4.1.1. Em ambas utilizações do C4.5 os parâmetros de poda para a construção da árvore, limiar de entropia (ϑ) e limiar de ganho de informação (τ), são sempre os mesmos e iguais a $\vartheta = 0.25$ e $\tau = 0$.

Segundo LEE em [29], a complexidade do classificador C4.5 é $O(i \cdot M \cdot log(M)) \mid 0 \leqslant i \leqslant N$, ou seja, $i = |S|$ onde S é o subconjunto avaliado.

implementação jaDTi esse caso não é tratado pois quando $F[\bullet] = 0, \ln(F[\bullet]) = -\infty \wedge F[\bullet] \times \ln(F[\bullet]) = 0$, o que não prejudica o funcionamento do algoritmo.

Algoritmo 3.10 Algoritmo para obtenção da precisão de uma Árvore de Decisão C4.5

ENTRADA: S, subconjunto que será avaliado
ENTRADA: Υ^{TR}, subconjunto de padrões para treinamento
ENTRADA: Υ^{T}, subconjunto de padrões para teste
SAÍDA: $\Psi(S)$

1: $AD \leftarrow C45(\Upsilon^{TR}, .25, 0)$; { Algoritmo 3.9 }
2: $certo \leftarrow 0$;
3: **PARA** $i \leftarrow 1, 2, \ldots, |\Upsilon^{T}|$ **FAÇA**
4: $\quad \rho \leftarrow \Upsilon^{T}[i]$;
5: $\quad$ **SE** $\Lambda(\rho) = AD(\rho)$ **ENTÃO**
6: $\quad\quad certo \leftarrow certo + 1$;
7: **RETORNE** $\frac{err}{|\Upsilon^{T}|}$;

3.3 Discretização de Características

Segundo LIU & MOTODA em [31], Discretização de Características é a tarefa que discretiza valores característicos em um número menor de intervalos onde cada intervalo é mapeado para um valor discreto. Alguns benefícios resultantes da aplicação deste processo incluem:

- Simplificar a descrição dos dados;
- Aumentar o entendimento dos dados bem como de qualquer resultado obtido por sistemas de Reconhecimento de Padrões;
- Possibilita que os dados possam ser aplicados a uma quantidade maior de algoritmos;
- Ameniza o Problema da Repetição na indução de árvores de decisão. No Problema da Repetição uma característica é testada repetidamente durante o percurso na árvore de decisão, BRODLEY & UTGOFF em [6].

Nas subseções seguintes estão descritos dois métodos simples (EWI e EFI) que realizam a discretização de forma automática. Existem outros métodos para Discretização de Características (DC), por exemplo, o ChiMerge que utiliza a métrica estatistica χ^2 para determinar se as frequências relativas das classes de intervalos adjacentes são distintamente diferentes ou se são suficientemente iguais para justificar a união dos dois intervalos em um único intervalo.

3.3.1 Intervalos de Tamanhos Iguais (EWI)

O método para DC Intervalos de Tamanhos Iguais (EWI, sigla para *Equal Width Intervals*) é o método utilizado neste trabalho para discretizar os valores característicos para possibilitar o uso da função critério Taxa de inconsistência (Subseção 3.2.1). Este método para DC separa os valores de uma característica em m intervalos entre os valores mínimo e máximo que a característica assume. O novo valor discreto associado é o intervalo ao qual o valor original pertence.

O Algoritmo 3.11, adaptado de LIU & MOTODA em [31], descreve a implementação utilizada para discretizar as bases de dados. Este algoritmo é aplicado no momento da instanciação do critéio Taxa de inconstência, é executado uma única vez. Nota-se que nenhum teste é realizado com relação à natureza dos dados, ele é aplicado mesmo às características que já são discretas. Esta implementação é totalmente automatizada, nenhum parâmetro adicional à matriz com os valores característicos é passado, m é calculado automaticamente (o tamanho de cada intervalo, $fator$, é igual ao menor valor em módulo diferente de zero que a característica assume, Linha 2). O valor discreto que é associado é o número inteiro que esteja mais próximo à razão entre o valor original e o tamanho do intervalo pelo uso do arredondamento, Linha 4.

Algoritmo 3.11 Algoritmo EWI

ENTRADA: Υ
SAÍDA: Υ'

1: **PARA** $i \leftarrow 1, 2, \ldots, N$ **FAÇA**
2: $\quad fator \leftarrow abs(c) \mid \forall v \in \Upsilon[][i], abs(c) \leqslant abs(v) \wedge c \neq 0$;
3: $\quad$ **PARA** $j \leftarrow 1, 2, \ldots, M$ **FAÇA**
4: $\quad\quad \Upsilon'[j][i] \leftarrow arredonda(\frac{\Upsilon[j][i]}{fator})$;

3.3.2 Intervalos de Frequências Iguais (EFI)

O método para DC Intervalos de Frequências Iguais (EFI, sigla para *Equal Frequency Intervals*) é outro método que discretiza valores característicos. Este método para DC encontra $m-1$ valores limites para m intervalos entre os valores mínimo e máximo, de modo que cada intervalo contenha aproximadamente a mesma quantidade de valores característicos. Este método não foi utilizado durante a execução deste trabalho[10].

[10] A implementação de outro processo discretizante obrigaria sua execução e comparação com os outros métodos o que demandaria um maior esforço computacional nos testes e mais uma fonte de comparação, que seria os

efeitos gerados pela discretização no processo para SC, o que foge ao escopo desta monografia. Caso este método discretizante fosse levado em consideração seis novos métodos deveriam ser avaliados, um para cada algoritmo integrado ao critério Taxa de Inconsistência, totalizando dezoito métodos ao todo.

Capítulo 4

Avaliação e Comparação Experimental

Este capítulo tem por objetivo definir a metodologia de comparação dos métodos para Seleção de Características (SC), bem como detalhar os testes executados sobre as bases de dados e os resultados obtidos. Os testes correspondem ao desempenho obtido em um computador com as seguintes configurações:

Processador AMD Sempron^{TM} 2.4MHz com aproximadamente 900MB de memória volátil;

Sistema Operacional Ubuntu 9.04 (Linux 2.6.28-15-generic) com GNOME 2.26.1;

Máquina Virtual Sun Java^{TM} *Runtime Environment* 6 (versão 1.6.0_16);

Parâmetrização Adicionalmente, a rotina atua atendendo as seguintes condições:

1. Executa em linha de comando, fora do ambiente gráfico (opção: Terminal de Segurança), para minimizar a disputa de recursos com outros processos;
2. Memória disponível para o processo que executa os testes é fixa e igual à 512MB[1];
3. Uso de Coletor de Lixo Incremental (*Incremental Garbage Colector*) que mantém um processo leve sempre em execução para liberar memória não utilizada.

4.1 Metodologia

A comparação é feita considerando o tempo de processamento necessário para a execução dos métodos[2], sobre as bases de dados definidas na Seção 4.2, e a qualidade da solução obtida. A

[1]Parte da memória estava em utilização por outros aplicativos básicos, poder-se-ia aumentar a quantidade de memória até um total de aproximadamente 700MB, o que poderia melhorar a performance dos métodos, porém optou-se por um valor equivalente a uma potência de 2.

[2]Os métodos são obtidos pela combinação de algoritmos e critérios.

qualidade da solução é quantificada pela precisão resultante da aplicação do classificador C4.5, que induz uma árvore de decisão, com diferentes subconjuntos de padrões para treinamento e para testes, (Subseção 4.1.1).

Para simplificar a notação, $método$ simboliza a identificação de um método qualquer, $algoritmo^{critério}$. Um método aplicado a uma base de dados possui precisão $ac_{base}^{método}$ ($ac^{método}$) e tempo de execução $ms_{base}^{método}$ ($ms^{método}$), com significado idêntico para $base$ que representa a identificação de uma base de dados qualquer. O critério Taxa de Inconsistência é denotado por CON.

Existem um total de 12 métodos, as combinações possíveis entre os seis algoritmos (SFS, SBS, SFFS, BAB', BAB e DEP) com os dois critérios (CON e C4.5), nos métodos sequenciais e no exaustivo todo o Espaço de Busca (EB) é utilizado, nos métodos de ramificação existem três testes com $n \approx \frac{N}{4}$, $n \approx \frac{N}{2}$ e $n \approx \frac{3 \cdot N}{4}$, EB é parcial, o que define os quatro grupos de comparação de acordo com o tamanho do EB. Essa divisão em grupos comparativos é devido às diferenças existentes entre os algoritmos, a exploração do EB pelos algoritmos de ramificação e poda depende da quantidade de características desejado, n. Cada grupo é comparado de forma independente. A comparação é realizada em duas etapas:

1. comparação local entre os métodos sobre cada base de dados, facilitando a apresentação dos testes e possibilitando que as quantidades de características N e as quantidades de padrões M diferentes em cada base de dados não interfiram nos resultados obtidos. Essa etapa expressa o comportamento dos métodos para a base de dados específica, os métodos podem ser mais ou menos vantajosos para cada base de dados;

2. comparação global entre os métodos, tem o objetivo de generalizar os resultados obtidos na etapa anterior para extrair o comportamento dos métodos de modo mais abrangente.

Na comparação local em cada base de dados para cada grupo comparativo, elege-se o método que induzir um classifcador com a melhor precisão, caso os classificadores induzidos por dois ou mais métodos obtenham a mesma precisão, então o tempo de processamento atua como critério de desempate. O método eleito é simbolizado por $\widetilde{método}$, com precisão ac e tempo de processamento ms, e é usado para calcular a taxa de ganho em precisão, com relação aos outros métodos, com a Equação 4.1 para cada metodo comparado, quanto maior melhor sendo

que os valores são compreendidos entre $(0, 1]$. A mesma lógica é empregada com relação ao tempo de execução pela Equação 4.2, porém, quanto menor melhor, caso a taxa de ganho em tempo seja menor que a unidade o tempo de execução do método analisado é menor que o tempo de execução do método eleito. A Equação 4.3 informa a taxa de ganho em precisão por tempo de execução em relação ao método pivô, mede o custo benefício do processamento, e é diretamente proporcional à taxa de ganho em precisão e inversamente proporcional à taxa de ganho em tempo. Na comparação local, os métodos são brevemente comparados com relação aos valores destas três taxas.

$$\widetilde{método}(ac, ms) \quad | \quad \forall método \in EXE \begin{cases} ac > ac^{método} & \text{, eleição} \\ ac \geqslant ac^{método} \wedge ms \leqslant ms^{método} & \text{, desempate} \end{cases}$$

$$ac(método) = \frac{ac^{método}}{ac}\text{, taxa de ganho em precisão} \tag{4.1}$$

$$ms(método) = \frac{ms^{método}}{ms}\text{, taxa de ganho em tempo} \tag{4.2}$$

$$av(método) = \frac{ac(método)}{ms(método)}\text{, taxa de ganho em precisão por ganho em tempo} \tag{4.3}$$

Na comparação global, as taxas de ganho de precisão por tempo calculadas em cada comparação local são agrupadas com relação ao método que representam, calcula-se a média (Equação 4.4), a variância (Equação 4.5) e o desvio padrão (Equação 4.6), onde q denota a quantidade de bases testadas. De posse de tais medidas estatísticas os métodos são confrontados dois a dois, com o uso da Equação 4.7, diferença absoluta em desvios padrões, para decidir se um método é melhor que outro: se $||método_A - método_B|| > 0$, o método A supera o B; se $||método_A - método_B|| \leqslant 0$, o método B supera o A. Esse tratamento estatístico foi adaptado de MONARD & BARANAUSKAS em [35] que o definem com o intuito de determinar a precisão de um classificador quando o conjunto de padrões para treinamento for estratificado, dividido em partes. Neste trabalho este tratamento estatístico é utilizado para estimar as taxas de ganhos em tempo, precisão e da razão de ambos para um número finito de execuções idênticas.

$$\overline{av}(método) = \frac{1}{q} \times \sum_{i=1}^{q} av(método^i) \tag{4.4}$$

$$var(método) = \frac{1}{q} \times \left(\frac{1}{q-1} \times \sum_{i=1}^{q} \left[av(método^i) - \overline{av}(método) \right]^2 \right) \tag{4.5}$$

$$\sigma(método) = \sqrt{var(método)} \tag{4.6}$$

$$\overline{av}(método_A - método_B) = \overline{av}(método_A) - \overline{av}(método_B)$$

$$\sigma(método_A - método_B) = \sqrt{\frac{[\sigma(método_A)]^2 + [\sigma(método_B)]^2}{2}}$$

$$||método_A - método_B|| = \frac{\overline{av}(método_A - método_B)}{\sigma(método_A - método_B)} \tag{4.7}$$

4.1.1 Avaliação de Soluções

MONARD & BARANAUSKAS em [35] aborda uma série de metodologias para avaliação de algoritmos para Aprendizado de Máquina, e entre elas:

Resubstituição Nesta metodologia, um mesmo conjunto de padrões é utilizado tanto para treinamento quanto para testes, o que não é uma prática aconselhável, pois a precisão obtida é aparente, tal que os testes que determinam a precisão do classificador inferido durante o treinamento ficam condicionados aos próprios padrões apresentados. Quando da utilização do C4.5 como critério, abordagem *wrapper* para SC (Seção 2.2 na Página 11), a resubstituição é utilizada. O motivo pela adoção da precisão aparente é manter o mesmo conjunto de padrões para os dois critérios (C4.5 e Taxa de Inconsistência), caso se reduzisse o conjunto de treinamento para ambos os critérios então a Taxa de Inconsistência seria afetada, porque o critério de Consistência atuaria apenas sobre uma parte dos dados. Diante deste impasse, o uso da precisão aparente para o critério C4.5 é a solução empregada;

Holdout Este estimador atua em base de dados divididas em duas partes, uma para treinamento e outra para teste, de acordo com uma porcentagem. A precisão é obtida sobre a classificação dos padrões para teste, conforme o Algoritmo 3.10 (Página 47). Neste trabalho, o estimador *holdout* é usado para avaliar os subconjuntos de características soluções obtidos na aplicação dos métodos para SC sobre as bases de dados divididas em aproximadamente $\frac{2}{3}$ para treinamento e $\frac{1}{3}$ para testes de forma aleatória.

Essa diferença na obtenção da precisão dos subconjuntos durante (critério com ressubstituição) e após o processo para SC (com *holdout* com o objetivo de avaliação) prejudica a qualidade

das soluções encontradas pelos métodos que utilizam o C4.5 como critério. O uso do classificador como métrica tem o objetivo de maximizar a precisão do classificador, contudo com estas diferenças a maximização da acuidade do classificador não é obtida, isso implica que na prática ambas avaliações do classificador C4.5 (com e sem *holdout*) se comportam como quantificadores diferentes. Portanto, a comparação se refere aos métodos om critérios Taxa de Inconsistência e do C4.5 com resubstituição, e o classificador C4.5 com estimador *holdout* serve para validar a qualidade das soluções obtidas pelos dois critérios.

4.2 Bases de Dados

As bases de dados utilizadas para os testes pertencem ao Repositório UCI (ASUNCION & NEWMAN em [4]). A escolha desta fonte de dados é devido à sua larga utilização na experimentação de soluções para Aprendizado de Máquinas e por conter uma quantidade considerável de bases de dados. Para a escolha das bases de dados foram considerados uma série de fatores: quantidade de características (N); quantidade de padrões (M); quantidades de classes (λ); distribuição de padrões por classe (uniforme, ou priorizando uma ou mais das classes devido à diferença entre as quantidades de amostras); tipo das características (discretas, contínuas ou ambas); qualidade das características (características são relevantes, redundantes ou irrelevantes). KUDO & SKLANSKY em [27] dizem que um problema para SC é pequeno quando $N \in (0, 19]$, médio quando $N \in [20, 49]$ e grande quando $N \in [50, \infty)$. Das dezoito base de dados escolhidas para a aplicação dos testes, metade são de pequeno porte, duas são de médio porte e as restantes de grande porte. As subseções subsequentes descrevem as bases de dados utilizadas para os testes, e seus resultados pela comparação pontual.

4.2.1 Iris

Base de dados criada por FISHER em [13] *apud* ASUNCION & NEWMAN em [4]. Contém 150 padrões em três classes (50 padrões por classes), uma classe é linearmente separável as outras duas não. Duas classes são linearmente separáveis quando a diferença entre elas for bem definida. Cada padrão é formado por quatro características, valores contínuos que represen-

tam respectivamente o tamanho e a largura da sépala[3] e da pétala[4] de uma flor ($sepalLength$, $sepalWidth$, $petalLength$ e $petalWidth$), mais a classe a qual a planta pertence (atributo alvo), cujos valores possíveis são $IrisSetosa$ (0), $IrisVersicolour$ (1), $IrisVirginica$ (2). O interesse sobre essa base de dados reside na dificuldade entre diferenciar (classificar) duas das classes. A Tabela 4.1 mostra o resultado dos testes realizados.

Tabela 4.1: Testes sobre a base de dados Iris

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	4	$48ms$	$0.98 : 3$	$0.9\overline{81}$
SBS	0	$42ms$	$0.98 : 3$	$0.9\overline{81}$
SFFS	4	$54ms$	$0.98 : 3$	$0.9\overline{81}$
BAB'	1	$63ms$	$0.98 : 3$	$0.9\overline{81}$
	2	$34ms$	$0.98 : 3$	$0.9\overline{81}$
	3	$29ms$	$0.98 : 3$	$0.9\overline{81}$
BAB	1	$70ms$	$0.98 : 3$	$0.9\overline{81}$
	2	$44ms$	$0.98 : 3$	$0.9\overline{81}$
	3	$28ms$	$0.98 : 3$	$0.9\overline{81}$
DEP	•	$46ms$	$0.98 : 3$	$0.9\overline{81}$
Classificador C4.5				
SFS	4	$329ms$	$0.98\overline{6} : 2$	$0.9\overline{63}$
SBS	0	$283ms$	$0.99\overline{3} : 3$	$0.9\overline{27}$
SFFS	4	$399ms$	$0.99\overline{3} : 3$	$0.9\overline{27}$
BAB'	1	$404ms$	$0.99\overline{3} : 3$	$0.\overline{90}$
	2	$288ms$	$0.99\overline{3} : 3$	$0.\overline{90}$
	3	$217ms$	$0.99\overline{3} : 3$	$0.\overline{90}$
BAB	1	$303ms$	$0.99\overline{3} : 3$	$0.\overline{90}$
	2	$288ms$	$0.99\overline{3} : 3$	$0.\overline{90}$
	3	$214ms$	$0.99\overline{3} : 3$	$0.\overline{90}$
DEP	•	$311ms$	$0.99\overline{3} : 3$	$0.\overline{90}$

A classificação com *holdout* para o conjunto com todas as características para a base de dados Iris resulta em uma precisão de $0.9\overline{81}$. A Tabela 4.2 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Para esta base de dados, levando em conta os valores das taxas presentes na Tabela 4.2, os melhores métodos foram os que utilizam como critério a Taxa de Inconsistência, a coluna ac da tabela assume o valor máximo (unidade) e coluna av valores próximos à unidade. Os

[3]Folhas modificadas com função de proteger o botão floral.

[4]Folhas modificadas e com uma série de funções, entre elas atração de polinizadores.

Tabela 4.2: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados Iris

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método} = SBS^{CON}$						
SFS	1	≈ 1.14286	0.875	≈ 0.98148	≈ 7.83333	≈ 0.1253
SBS	1	1	1	≈ 0.94444	≈ 6.7381	≈ 0.14016
SFFS	1	≈ 1.28571	≈ 0.77778	≈ 0.94444	9.5	≈ 0.09942
DEP	1	≈ 1.09524	≈ 0.91304	≈ 0.92593	≈ 7.40476	≈ 0.12504
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.92593	≈ 6.4127	≈ 0.14439
BAB	1	≈ 1.11111	0.9	≈ 0.92593	≈ 4.80952	≈ 0.19252
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.92593	≈ 8.47059	≈ 0.10931
BAB	1	≈ 1.29412	≈ 0.77273	≈ 0.92593	≈ 8.47059	≈ 0.10931
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método} = BAB^{CON}$						
BAB'	1	≈ 1.03571	≈ 0.96552	≈ 0.92593	7.75	≈ 0.11947
BAB	1	1	1	≈ 0.92593	≈ 7.64286	≈ 0.12115

métodos que utilizam o classificador C4.5, todos obtiveram desempenho inferior, coluna ms contém valores muito altos, apesar de encontrarem subconjuntos de características razoáveis, ainda que inferiores aos encontrados por métodos com Taxa de Inconsistência.

4.2.2 *Blood Transfusion Service Center*

Base de dados criada por YEH, YANG & TING em [61] *apud* ASUNCION & NEWMAN em [4]. Contém 748 amostras divididas em duas classes, problema binário, que indicam se uma pessoa doou ou não sangue em março de 2007, 570 não doaram. Todas as características são discretas, são elas: R, meses desde a última doação; F, quantidade de doações; M, total de sangue doado em c.c. e T, meses desde a primeira doação. Por conter a mesma quantidade de características da base de dados Iris, essa base de dados mostra indícios do comportamento dos métodos quanto à quantidade de padrões (M). A Tabela 4.3 mostra o resultado dos testes realizados.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Blood Transfusion Service Center* resulta em uma precisão de ≈ 0.74. A Tabela 4.4 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por

Tabela 4.3: Testes sobre a base de dados *Blood Transfusion Service Center*

Algoritmo	n	**Tempo**	$\Psi(S_\oplus) : \lvert S_\oplus \rvert$	$precisão(S_\oplus, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	4	$61ms$	$0.93\overline{18} : 3$	≈ 0.74
SBS	0	$63ms$	$0.93\overline{18} : 3$	≈ 0.74
SFFS	4	$81ms$	$0.93\overline{18} : 3$	≈ 0.74
BAB'	1	$67ms$	$0.93\overline{18} : 3$	≈ 0.74
	2	$67ms$	$0.93\overline{18} : 3$	≈ 0.74
	3	$58ms$	$0.93\overline{18} : 3$	≈ 0.74
BAB	1	$67ms$	$0.93\overline{18} : 3$	≈ 0.74
	2	$64ms$	$0.93\overline{18} : 3$	≈ 0.74
	3	$58ms$	$0.93\overline{18} : 3$	≈ 0.74
DEP	•	$65ms$	$0.93\overline{18} : 3$	≈ 0.74
Classificador C4.5				
SFS	4	$639ms$	$\approx 0.9291 : 3$	≈ 0.74
SBS	0	$723ms$	$\approx 0.9291 : 3$	≈ 0.74
SFFS	4	$1s290ms$	$\approx 0.9291 : 3$	≈ 0.74
BAB'	1	$819ms$	$\approx 0.9291 : 3$	≈ 0.74
	2	$776ms$	$\approx 0.9291 : 3$	≈ 0.74
	3	$581ms$	$\approx 0.9291 : 3$	≈ 0.74
BAB	1	$812ms$	$\approx 0.9291 : 3$	≈ 0.74
	2	$762ms$	$\approx 0.9291 : 3$	≈ 0.74
	3	$572ms$	$\approx 0.9291 : 3$	≈ 0.74
DEP	•	$827ms$	$\approx 0.9291 : 3$	≈ 0.74

tempo.

Analisando a Tabela 4.4, nota-se que todos os métodos, para todos os grupos de comparação encontraram boas soluções, coluna (ac) tem valor unitário, porém, os métodos que utilizam o classificador C4.5 obtiveram valores para a taxa de ganho (perda) em tempo de 9 a 21 vezes maiores que os métodos utilizando a Taxa de Inconsistência para encontrar a solução.

4.2.3 *E. coli*

Contém 336 padrões em 8 classes que informam a localização de proteínas em células *E. coli*. As classes e a quantidade de padrões por classe são: cp (0), citoplasma com 143; im (1), membrana interna, sem sinal de sequência, com 77; pp (2), periplasma com 52; imU (3), membrana interna, com sinal de sequência não passível a decomposição[5], com 35; om (4), membrana externa com 20; omL (5), lipoproteína de membrana externa com 5; imL (6), lipoproteína de

[5]Tradução adotada para *inner membrane, uncleavable signal sequence*

Tabela 4.4: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Blood Transfusion Service Center*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método} = SFS^{CON}$						
SFS	1	1	1	1	≈ 10.47541	≈ 0.09546
SBS	1	≈ 1.03279	≈ 0.96825	1	≈ 11.85246	≈ 0.08437
SFFS	1	≈ 1.32787	≈ 0.75309	1	≈ 21.14754	≈ 0.04729
DEP	1	≈ 1.06557	≈ 0.93846	1	≈ 13.55738	≈ 0.07376
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método} = BAB^{CON}$						
BAB'	1	1	1	1	≈ 12.22388	≈ 0.08181
BAB	1	1	1	1	≈ 12.1194	≈ 0.08251
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método} = BAB^{CON}$						
BAB'	1	≈ 1.04688	≈ 0.95522	1	≈ 12.125	≈ 0.08247
BAB	1	1	1	1	≈ 11.90625	≈ 0.08399
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método} = BAB^{CON}$						
BAB'	1	1	1	1	≈ 10.01724	≈ 0.09983
BAB	1	1	1	1	≈ 9.86207	≈ 0.1014

membrana interna com 2; e imS (7), membrana interna, com sinal de sequência passível a decomposição[6], com 2 padrões. As características[7] são identificadas por mcg, gvh, lip, chg, aac, $alm1$ e $alm2$ que representam coeficientes preditivos obtidos por diferentes métodos. Exceto pelos atributos lip e chg que são binários, todos os atributos são contínuos. Como é formada por atributos preditivos, esta base de dados é de interesse por não apresentar atributos irrelevantes e por estar relacionada à base Yeast, além de conter quantidades diferentes de padrões para cada classe. A Tabela 4.5 mostra o resultado dos testes realizados.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *E. coli* resulta em uma precisão de 0.775. A Tabela 4.6 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

De acordo com os dados presentes na Tabela 4.6, os métodos com Taxa de Inconsistência foram superiores para esta base de dados, por um fator que varia de aproximadamente 9 a 38 vezes a taxa de ganho em tempo. Entre os métodos do grupo que explora todo o EB e utilizam Taxa de Inconsistência, nota-se que o mélhor método foi o de Busca Exaustiva, todavia, os

[6]Tradução adotada para *inner membrane, cleavable signal sequence*

[7]Originalmente, continha oito características onde uma, a primeira, era um nome sequencial para identificar a amostra, tal característica foi removida previamente para impedir sua influência no cômputo dos critérios.

Tabela 4.5: Testes sobre a base de dados *E. coli*

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	7	$64ms$	$\approx 0.997 : 4$	≈ 0.83
SBS	0	$62ms$	$\approx 0.997 : 4$	≈ 0.83
SFFS	7	$94ms$	$\approx 0.997 : 4$	≈ 0.83
BAB'	2	$94ms$	$\approx 0.997 : 4$	$0.8\overline{6}$
	4	$59ms$	$\approx 0.997 : 4$	$0.8\overline{6}$
	6	$48ms$	$\approx 0.997 : 6$	$0.8\overline{6}$
BAB	2	$94ms$	$\approx 0.997 : 4$	$0.8\overline{6}$
	4	$62ms$	$\approx 0.997 : 4$	$0.8\overline{6}$
	6	$46ms$	$\approx 0.997 : 6$	$0.8\overline{6}$
DEP	•	$106ms$	$\approx 0.997 : 4$	$0.8\overline{6}$
Classificador C4.5				
SFS	7	$1s048ms$	$\approx 0.997 : 3$	$0.741\overline{6}$
SBS	0	$1s321ms$	$\approx 0.991 : 4$	≈ 0.7083
SFFS	7	$2s370ms$	$\approx 0.997 : 3$	$0.741\overline{6}$
BAB'	2	$2s816ms$	$\approx 0.997 : 3$	$0.741\overline{6}$
	4	$771ms$	$\approx 0.991 : 4$	$0.8\overline{6}$
	6	$665ms$	$\approx 0.991 : 6$	$0.8\overline{6}$
BAB	2	$2s825ms$	$\approx 0.997 : 3$	$0.741\overline{6}$
	4	$1s330ms$	$\approx 0.991 : 4$	$0.8\overline{6}$
	6	$645ms$	$\approx 0.991 : 6$	$0.8\overline{6}$
DEP	•	$4s092ms$	$\approx 0.997 : 3$	$0.741\overline{6}$

três métodos sequenciais encontraram solução em menos tempo, o que elevou a taxa de custo benefício deste métodos, coluna av com valores maiores que a unidade.

4.2.4 Yeast

Contém 1484 padrões divididos em 10 classes que informam a localização de proteínas em células eucariontes[8]. As classes e a quantidade de padrões por classe são: CYT (0), citosólico ou citoesquelético com 463; NUC (1), nuclear com 429; MIT (2), mitocondrial com 244; ME3 (3), menbrana proteica, sem sinal N-terminal[9], com 163; ME2 (4), menbrana proteica, com sinal não passível a decomposição[10], com 51; ME1 (5), menbrana proteica, com sinal passível a decomposição[11], com 44; EXC (6), extracelular com 35; VAC (7), vacuolar com 30; POX

[8]Células com núcleo bem determinado pela presença de uma membrana específica, carioteca.
[9]Tradução adotada para *membrane protein, no N-terminal signal*
[10]Tradução adotada para *membrane protein, uncleaved signal*
[11]Tradução adotada para *membrane protein, cleaved signal*

Tabela 4.6: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *E. coli*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método} = DEP^{CON}$						
SFS	≈ 0.96154	≈ 0.60377	≈ 1.59255	≈ 0.85577	≈ 9.88679	≈ 0.08656
SBS	≈ 0.96154	≈ 0.58491	≈ 1.64392	≈ 0.81731	≈ 12.46226	≈ 0.06558
SFFS	≈ 0.96154	≈ 0.88679	≈ 1.08429	≈ 0.85577	≈ 22.35849	≈ 0.03827
DEP	1	1	1	≈ 0.85577	≈ 38.60377	≈ 0.02217
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.85577	≈ 29.95745	≈ 0.02857
BAB	1	1	1	≈ 0.85577	≈ 30.05319	≈ 0.02848
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	1	≈ 13.0678	≈ 0.07652
BAB	1	≈ 1.05085	≈ 0.81436	1	≈ 22.54237	≈ 0.04436
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método} = BAB^{CON}$						
BAB'	1	≈ 1.04348	≈ 0.95833	1	≈ 14.45652	≈ 0.06917
BAB	1	1	1	1	≈ 14.02174	≈ 0.07132

(8), peroxisomal com 20; e ERL (9), lúmen do retículo endoplasmático com 5 padrões. As características[12] são identificadas por mcg, gvh, alm, mit, erl, pox, vac e nuc que representam coeficientes preditivos obtidos por diferentes métodos. Exceto pelo atributo erl que é binário, todos os atributos são contínuos. Como é formada por atributos preditivos, esta base de dados é de interesse por não apresentar atributos irrelevantes e por estar relacionada à base E.coli, além de conter quantidades diferentes de padrões por classe. A Tabela 4.7 mostra o resultado dos testes realizados.

A classificação com *holdout* para o conjunto com todas as características para a base de dados Yeast resulta em uma precisão de ≈ 0.44. A Tabela 4.8 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Pela Tabela 4.8, apesar dos métodos com critério C4.5 encontrarem soluções razoáveis, o custo de execução destes métodos nesta base de dados, ms com valores muito elevados, fez com que a taxa de ganho em precisão por tempo para quase a totalidade dos métodos fosse reduzida a valores centesimais.

[12]Originalmente, continha nove características onde uma, a primeira, era um nome sequencial para identificar a amostra, tal característica foi removida previamente para impedir sua influência no cômputo dos critérios.

Tabela 4.7: Testes sobre a base de dados Yeast

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	8	$138ms$	$\approx 0.85 : 8$	≈ 0.44
SBS	0	$148ms$	$\approx 0.85 : 8$	≈ 0.44
SFFS	8	$333ms$	$\approx 0.85 : 8$	≈ 0.44
BAB'	2	$499ms$	$\approx 0.85 : 8$	≈ 0.44
	4	$337ms$	$\approx 0.85 : 8$	≈ 0.44
	6	$90ms$	$\approx 0.85 : 8$	≈ 0.44
BAB	2	$497ms$	$\approx 0.85 : 8$	≈ 0.44
	4	$339ms$	$\approx 0.85 : 8$	≈ 0.44
	6	$89ms$	$\approx 0.85 : 8$	≈ 0.44
DEP	•	$770ms$	$\approx 0.85 : 8$	≈ 0.44
Classificador C4.5				
SFS	8	$9s033ms$	$1.0 : 4$	≈ 0.42
SBS	0	$13s443ms$	$1.0 : 4$	≈ 0.42
SFFS	8	$43s973ms$	$1.0 : 4$	≈ 0.42
BAB'	2	$44s457ms$	$1.0 : 4$	≈ 0.42
	4	$17s317ms$	$1.0 : 4$	≈ 0.42
	6	$6s418ms$	$1.0 : 6$	≈ 0.39
BAB	2	$44s484ms$	$1.0 : 4$	≈ 0.42
	4	$16s926ms$	$1.0 : 4$	≈ 0.42
	6	$6s354ms$	$1.0 : 6$	≈ 0.39
DEP	•	$1min14s441ms$	$1.0 : 4$	≈ 0.42

4.2.5 *Shuttle Land Control*

Contém 58000 padrões em sete classes designadas por 0 a 6 (*Rad Flow*, *Fpv Close*, *Fpv Open*, *High*, *Bypass*, *Bpv Close*, *Bpv Open*)[13]. Cada padrão é formado por 9 características discretas. Por conter uma grande quantidade de padrões e por 80% destes padrões pertencem a uma mesma classe, a primeira, esta base de dados é peculiar é de interesse para o estudo. A Tabela 4.9 mostra o resultado dos testes realizados.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Shuttle Land Control* resulta em uma precisão de ≈ 0.99453. A Tabela 4.10 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Esta base de dados contém uma quantidade excessiva de padrões ($M = 58000$), e oferece indícios do comportamento dos métodos quando aplicados a situações extremas onde a quanti-

[13]Originalmente eram identificadas de 1 a 7. Essa mudança é devido aos pré-requisitos da implementação utilizada para o classificador C4.5.

Tabela 4.8: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados Yeast

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método} = SFS^{CON}$						
SFS	1	1	1	≈ 0.95633	≈ 65.45652	≈ 0.01461
SBS	1	≈ 1.07246	≈ 0.93243	≈ 0.95633	≈ 97.41304	≈ 0.00982
SFFS	1	≈ 2.41304	≈ 0.41441	≈ 0.95633	≈ 318.64493	≈ 0.003
DEP	1	≈ 5.57971	≈ 0.17922	≈ 0.95633	≈ 539.42754	≈ 0.00177
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método} = BAB^{CON}$						
BAB'	1	≈ 1.00402	≈ 0.99599	≈ 0.95633	≈ 89.4507	≈ 0.01069
BAB	1	1	1	≈ 0.95633	≈ 89.50503	≈ 0.01068
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.95633	≈ 51.38576	≈ 0.01861
BAB	1	≈ 1.00593	≈ 0.9941	≈ 0.95633	≈ 50.22552	≈ 0.01904
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método} = BAB^{CON}$						
BAB'	1	≈ 1.01124	≈ 0.98889	≈ 0.87773	≈ 72.11236	≈ 0.01217
BAB	1	1	1	≈ 0.87773	≈ 71.39326	≈ 0.01229

dade de amostras for muito elevada. Na Tabela 4.10, os métodos com critério C4.5 obtiveram os piores desempenhos, mas, é de interesse observar também que os métodos com critérios Taxa de Inconsistência (sequenciais regressivo, flutuante e de busca exaustiva no primeiro grupo, e os métodos BAB originais nos dois últimos grupos) demoraram bastante para obter as soluções.

4.2.6 *Glass Identification*

Contém 214 padrões em 6 classes que correspondem a tipos de vidros, identificados por: 0 (*building windows float processed*, 70 padrões), 1 (*building windows non float processed*, 76 padrões), 2 (*vehicle windows float processed*, 17 padrões), 3 (*containers*, 13 padrões), 4 (*tableware*, 9 padrões), 5 (*headlamps*, 29 padrões)[14], duas classes predominam o que é de interesse. Cada padrão é formado por nove características[15] representando quantidades de constituintes químicos presentes nas amostras mais um índice de refrativo. As características são: RI (índice refrativo), Na (sódio), Mg (magnésio), Al (alumínio), Si (silício), K (potásio), Ca (cálcio) e Ba (bário), todas as características são contínuas. A Tabela 4.11 mostra o resultado

[14]Originalmente eram identificados de 1 até 7. Sendo que a classe com valor 4 (*vehicle windows non float processed*) não ocorre (sem padrões), motivo pelo qual foi removida nesse estudo.

[15]Originalmente dez, a primeira era um identificador numérico e foi removido.

Tabela 4.9: Testes sobre a base de dados *Shuttle Land Control*

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	9	$5s708ms$	$1.0 : 4$	≈ 0.99458
SBS	0	$8s154ms$	$1.0 : 4$	≈ 0.99153
SFFS	9	$18s296ms$	$1.0 : 4$	≈ 0.99458
BAB'	2	$35s228ms$	$1.0 : 4$	≈ 0.99205
	5	$9s904ms$	$1.0 : 5$	≈ 0.99458
	7	$3s189ms$	$1.0 : 7$	≈ 0.99458
BAB	2	$35s212ms$	$1.0 : 4$	≈ 0.99205
	5	$40s030ms$	$1.0 : 5$	≈ 0.99458
	7	$9s407ms$	$1.0 : 7$	≈ 0.99458
DEP	•	$1min13s741ms$	$1.0 : 4$	≈ 0.99458
Classificador C4.5				
SFS	9	$2min29s246ms$	$\approx 0.99469 : 2$	≈ 0.99458
SBS	0	$3min17s107ms$	$\approx 0.99469 : 2$	≈ 0.99458
SFFS	9	$7min40s821ms$	$\approx 0.99469 : 2$	≈ 0.99458
BAB'	2	$17min05s119ms$	$\approx 0.99617 : 3$	≈ 0.99148
	5	$2min41s651ms$	$\approx 0.99469 : 5$	≈ 0.99458
	7	$1min18s160ms$	$\approx 0.99469 : 7$	≈ 0.99458
BAB	2	$16min41s980ms$	$\approx 0.99617 : 3$	≈ 0.99148
	5	$2min39s052ms$	$\approx 0.99469 : 5$	≈ 0.99458
	7	$1min16s142ms$	$\approx 0.99469 : 7$	≈ 0.99458
DEP	•	$41min45s205ms$	$\approx 0.99617 : 3$	≈ 0.99148

dos testes realizados.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Glass Identification* resulta em uma precisão de 0.68. A Tabela 4.12 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Nesta base de dados, para o primeiro e para o último grupos da Tabela 4.12, os métodos pivôs utilizam o classificador C4.5, e foram superados pelos métodos com Taxa de Inconsistência na medida de custo e benefício somente porque os tempos execução foram muito elevados.

Tabela 4.10: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Shuttle Land Control*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método = SFS^{CON}}$						
SFS	1	1	1	1	≈ 26.14681	≈ 0.03825
SBS	≈ 0.99694	≈ 1.42852	≈ 0.69788	1	≈ 34.53171	≈ 0.02896
SFFS	1	≈ 3.20533	≈ 0.31198	1	≈ 80.73248	≈ 0.01239
DEP	1	≈ 12.91889	≈ 0.07741	≈ 0.99689	≈ 438.89366	≈ 0.00227
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método = BAB^{CON}}$						
BAB'	1	≈ 1.00045	≈ 0.99955	≈ 0.99943	≈ 29.11277	≈ 0.03433
BAB	1	1	1	≈ 0.99943	≈ 28.45564	≈ 0.03512
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 16.32179	≈ 0.06127
BAB	1	≈ 4.0418	≈ 0.24741	1	≈ 16.05937	≈ 0.06227
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 24.50925	≈ 0.0408
BAB	1	≈ 2.94983	≈ 0.339	1	≈ 23.87645	≈ 0.04188

Tabela 4.11: Testes sobre a base de dados *Glass Identification*

Algoritmo	n	**Tempo**	$\Psi(S_\oplus) : \lvert S_\oplus \rvert$	$precisão(S_\oplus, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	9	$76ms$	$\approx 0.98 : 7$	0.64
SBS	0	$69ms$	$\approx 0.98 : 7$	0.64
SFFS	9	$95ms$	$\approx 0.98 : 7$	0.64
BAB'	2	$157ms$	$\approx 0.98 : 7$	0.64
	5	$93ms$	$\approx 0.98 : 7$	0.64
	7	$51ms$	$\approx 0.98 : 7$	0.64
BAB	2	$137ms$	$\approx 0.98 : 7$	0.64
	5	$95ms$	$\approx 0.98 : 7$	0.64
	7	$50ms$	$\approx 0.98 : 7$	0.64
DEP	•	$262ms$	$\approx 0.98 : 7$	0.64
Classificador C4.5				
SFS	9	$1s267ms$	$1.0 : 2$	≈ 0.49
SBS	0	$1s456ms$	$1.0 : 3$	$0.\overline{6}$
SFFS	9	$3s699ms$	$1.0 : 2$	≈ 0.49
BAB'	2	$1s220ms$	$1.0 : 2$	$0.58\overline{6}$
	5	$863ms$	$1.0 : 5$	0.56
	7	$768ms$	$1.0 : 7$	$0.65\overline{3}$
BAB	2	$1s805ms$	$1.0 : 2$	$0.58\overline{6}$
	5	$1s056ms$	$1.0 : 5$	0.56
	7	$770ms$	$1.0 : 7$	$0.65\overline{3}$
DEP	•	$11s072ms$	$1.0 : 2$	≈ 0.49

Tabela 4.12: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Glass Identification*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $método = SBS^{C4.5}$						
SFS	0.96	≈ 0.0522	≈ 18.39158	0.74	≈ 0.87019	≈ 0.85039
SBS	0.96	≈ 0.04739	≈ 20.25739	1	1	1
SFFS	0.96	≈ 0.06525	≈ 14.71326	0.74	≈ 2.54052	≈ 0.29128
DEP	0.96	≈ 0.17995	≈ 5.33496	0.74	≈ 7.6044	≈ 0.09731
EB parcial $n \approx \frac{N}{4}$, $método = BAB^{CON}$						
BAB'	1	≈ 1.14599	≈ 0.87261	≈ 0.91667	≈ 8.90511	≈ 0.10294
BAB	1	1	1	≈ 0.91667	≈ 13.17518	≈ 0.06958
EB parcial $n \approx \frac{N}{2}$, $método = BAB^{CON}$						
BAB'	1	1	1	0.875	≈ 9.27957	≈ 0.09429
BAB	1	≈ 1.02151	≈ 0.97895	0.875	≈ 11.35484	≈ 0.07706
EB parcial $n \approx \frac{3 \cdot N}{4}$, $método = BAB^{C4.5}$						
BAB'	≈ 0.97959	≈ 0.06641	≈ 14.7515	1	1	1
BAB	≈ 0.97959	≈ 0.0651	≈ 15.04653	1	≈ 1.0026	≈ 0.9974

4.2.7 *Wine*

Contém 178 padrões em três classes (59, 71 e 48 padrões respectivamente para as classes 0, 1 e 2)[16]. Cada padrão é formado por treze características que representam constituintes químicos ($alcohol$, $malicAcid$, ash, $alcalinityOfAsh$, $magnesium$, $totalPhenols$, $flavanoids$, $nonflavanoidPhenols$, $proanthocyanins$, $colorIntensity$, hue, $OD280/OD315.ofDilutedWines$ e $proline$) presentes em vinhos de três diferentes cultivares, apenas as características $magnesium$ e $proline$ são discretas, as restantes são contínuas. Dentro do contexto da classificação esta base de dados não oferece grandes dificuldades e um processo para SC sobre estes dados não deve deteriorar a precisão. A Tabela 4.13 mostra o resultado dos testes realizados.

Tabela 4.13: Testes sobre a base de dados *Wine*

Algoritmo	n	**Tempo**	$\Psi(S_\oplus) : \lvert S_\oplus \rvert$	$precisão(S_\oplus, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	13	$74ms$	$1.0 : 3$	$0.7\overline{42}$
SBS	0	$76ms$	$1.0 : 4$	$0.9\overline{24}$
SFFS	13	$108ms$	$1.0 : 3$	$0.7\overline{42}$
BAB'	3	$165ms$	$1.0 : 3$	$0.\overline{84}$
	7	$61ms$	$1.0 : 7$	$0.\overline{90}$
	10	$52ms$	$1.0 : 10$	$0.\overline{87}$
BAB	3	$608ms$	$1.0 : 3$	$0.\overline{84}$
	7	$983ms$	$1.0 : 7$	$0.\overline{90}$
	10	$207ms$	$1.0 : 10$	$0.\overline{87}$
DEP	•	$1s715ms$	$1.0 : 3$	$0.\overline{81}$
Classificador C4.5				
SFS	13	$1s118ms$	$\approx 0.994 : 2$	$0.\overline{78}$
SBS	0	$1s234ms$	$1.0 : 3$	$0.8\overline{63}$
SFFS	13	$2s315ms$	$\approx 0.994 : 2$	$0.\overline{78}$
BAB'	3	$1s014ms$	$1.0 : 4$	$0.\overline{93}$
	7	$1s071ms$	$\approx 0.99 : 7$	$0.8\overline{63}$
	10	$591ms$	$\approx 0.99 : 10$	≈ 0.83
BAB	3	$1s014ms$	$1.0 : 4$	$0.\overline{93}$
	7	$1s474ms$	$\approx 0.994 : 7$	$0.\overline{90}$
	10	$692ms$	$\approx 0.994 : 10$	$0.\overline{87}$
DEP	•	$1min05s987ms$	$1.0 : 2$	$0.\overline{60}$

A classificação com *holdout* para o conjunto com todas as características para a base de

[16] As classes eram originalmente identificadas por 1, 2 e 3, originalmente na primeira coluna dos dados.

dados *Wine* resulta em uma precisão de $0.\overline{93}$. A Tabela 4.14 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Tabela 4.14: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Wine*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $método = SBS^{CON}$						
SFS	≈ 0.80328	≈ 0.97368	≈ 0.82499	≈ 0.85246	≈ 14.71053	≈ 0.05795
SBS	1	1	1	≈ 0.93443	≈ 16.23684	≈ 0.05755
SFFS	≈ 0.80328	≈ 1.42105	≈ 0.56527	≈ 0.85246	≈ 30.46053	≈ 0.02799
DEP	≈ 0.88525	≈ 22.56579	≈ 0.03923	≈ 0.65574	≈ 868.25	≈ 0.00076
EB parcial $n \approx \frac{N}{4}$, $método = BAB'^{C4.5}$						
BAB'	≈ 0.90323	≈ 0.16272	≈ 5.55073	1	1	1
BAB	≈ 0.90323	≈ 0.59961	≈ 1.50637	1	1	1
EB parcial $n \approx \frac{N}{2}$, $método = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.95	≈ 17.55738	≈ 0.05411
BAB	1	≈ 16.11475	≈ 0.06205	1	≈ 24.16393	≈ 0.04138
EB parcial $n \approx \frac{3 \cdot N}{4}$, $método = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.94828	≈ 11.36538	≈ 0.08344
BAB	1	≈ 3.98077	≈ 0.25121	1	≈ 13.30769	≈ 0.07514

Para esta base de dados, os dados da Tabela 4.14 referentes aos três grupos de ramificação e poda, os métodos BAB' com critério monotônico obtiveram valores superiores com relação à métrica custo benefício. Nota-se que o custo de execução dos métodos exaustivos é muito superior ao dos custos dos métodos sequenciais.

4.2.8 *Letter Recognition*

Contém 20000 padrões em 26 classes, uma para cada letra maiúscula do alfabeto (respectivamente com 789, 766, 736, 805, 768, 775, 773, 734, 755, 747, 739, 761, 792, 783, 753, 803, 783, 758, 748, 796, 813, 764, 752, 787, 786 e 734 padrões). São dezesseis características discretas que correspondem a momentos estatísticos e contagens de borda, são elas: $Xbox$, $Ybox$, $width$, $high$, $onPix$, $Xbar$, $Ybar$, $X2bar$, $Y2bar$, $XYbar$, $X2Ybr$, $XY2br$, $Xege$, $XegvY$, $Yege$ e $YegvX$. O interesse nesta base de dados é devido à quantidade de padrões e a distribuição uniforme entre as classes. A Tabela 4.15 mostra o resultado dos testes realizados[17].

Tabela 4.15: Testes sobre a base de dados *Letter Recognition*

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	16	$7s377ms$	$1.0 : 12$	≈ 0.8568
SBS	0	$10s066ms$	$1.0 : 12$	≈ 0.8729
SFFS	16	$28s641ms$	$1.0 : 11$	≈ 0.8614
BAB'	4	$13min43s129ms$	$1.0 : 11$	≈ 0.8607
	8	$23min08s871ms$	$1.0 : 11$	≈ 0.8607
	12	$16s389ms$	$1.0 : 12$	≈ 0.8507
BAB	4	$13min46s295ms$	$1.0 : 11$	≈ 0.8607
	8	$23min21s894ms$	$1.0 : 11$	≈ 0.8607
	12	$1min08s349ms$	$1.0 : 12$	≈ 0.8507
DEP	•	$1h10min07s096ms$	$1.0 : 11$	≈ 0.8535
Classificador C4.5				
SFS	16	$33min43s480ms$	$0.9927 : 12$	≈ 0.8541
SBS	0	$56min08s502ms$	$0.9927 : 11$	≈ 0.8692
SFFS	16	$2h51min01s625ms$	$0.99305 : 11$	≈ 0.8544
BAB'	4	$3d29min53s563ms$	$0.99305 : 11$	≈ 0.8544
	8	$2hs53min30s129ms$	$0.9927 : 12$	≈ 0.8541
	12	$21min09s849ms$	$0.9926 : 13$	≈ 0.8522
BAB	4	$3d20min13s295ms$	$0.99305 : 11$	≈ 0.8544
	8	$2h54min31s400ms$	$0.9927 : 12$	≈ 0.8541
	12	$21min03s844ms$	$0.9926 : 13$	≈ 0.8522

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Letter Recognition* resulta em uma precisão de ≈ 0.8683. A Tabela 4.16 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

[17] A execução do método DEP$^{C4.5}$ não pôde ser completada devido a uma queda de tensão depois de aproximadamente uma semana de execução.

Na Tabela 4.16, no primeiro grupo, os métodos SFS^{CON} e SBS^{CON} se destacam por obterem os melhores valores para as taxas de ganho em precisão, tempo e de custo benefício. Nos grupos que ramificam e podam, a monotonicidade foi decisiva e obteve melhores resultados e com menor custo se comparado com o outro critério.

4.2.9 *Image Segmentation*

Contém 2310 padrões em sete classes cada qual com 330 padrões: $BRICKFACE$ (0), tijolos; SKY (1), céu; $FOLIAGE$ (2), folhagem; $CEMENT$ (3), cimento; $WINDOW$ (4), janela; $PATH$ (5), caminho; e $GRASS$ (6), capim. Cada padrão é formado por 19 características: $regionCentroidCol$, $regionCentroidRow$, $regionPixelCount$, $shortLineDensity5$, $shortLineDensity2$, $vedgeMean$, $vegdeSD$, $hedgeMean$, $hedgeSD$, $intensityMean$, $rawRedMean$, $rawBlueMean$, $rawGreenMean$, $exRedMean$, $exBlueMean$, $exGreenMean$, $valueMean$, $saturationMean$ e $hueMean$. A escolha desta base é devido à distribuição uniforme dos padrões entre as classes. A Tabela 4.17 mostra o resultado dos testes realizados.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Image Segmentation* resulta em uma precisão de ≈ 0.951. A Tabela 4.18 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Observando os valores de custo benefício na Tavela 4.18, é visível que os algoritmos sequenciais em oposição aos exaustivos, no primeiro grupo, e os métodos BAB' com critério monotônico, em oposição aos outros métodos nos demais grupos são os métodos que apresentam os melhores comportamentos para a base de dados *Image Segmentation*.

Tabela 4.16: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Letter Recognition*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método = SBS^{CON}}$						
SFS	≈ 0.98151	≈ 0.73286	≈ 1.33928	≈ 0.97842	≈ 201.02126	≈ 0.00487
SBS	1	1	1	≈ 0.99572	≈ 334.64157	≈ 0.00298
SFFS	≈ 0.98682	≈ 2.84532	≈ 0.34682	≈ 0.97877	≈ 1019.43423	≈ 0.00096
DEP	≈ 0.97774	≈ 417.95112	≈ 0.00234	*	*	*
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	≈ 0.99271	≈ 317.07492	≈ 0.00313
BAB	≈ 0.99271	≈ 1.00385	≈ 0.9889	≈ 0.99271	≈ 316.36997	≈ 0.00314
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	≈ 0.99236	≈ 4.38495	≈ 0.22631
BAB	1	≈ 1.00938	≈ 0.99071	≈ 0.99236	≈ 4.42907	≈ 0.22406
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método = BAB'^{C4.5}}$						
BAB'	≈ 0.99825	≈ 0.01297	≈ 76.98012	1	≈ 1.00475	≈ 0.99527
BAB	≈ 0.99825	≈ 0.05408	≈ 18.4586	1	1	1

Tabela 4.17: Testes sobre a base de dados *Image Segmentation*

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	19	$882ms$	$1.0 : 3$	≈ 0.655
SBS	0	$1s132ms$	$1.0 : 3$	≈ 0.868
SFFS	19	$2s098ms$	$1.0 : 3$	≈ 0.655
BAB'	5	$3s098ms$	$1.0 : 5$	≈ 0.886
	10	$1s218ms$	$1.0 : 10$	≈ 0.911
	15	$399ms$	$1.0 : 15$	≈ 0.896
BAB	5	$5min44s554ms$	$1.0 : 5$	≈ 0.886
	10	$17min32s473ms$	$1.0 : 10$	≈ 0.911
	15	$29s972ms$	$1.0 : 15$	≈ 0.896
DEP	•	$37min40s232ms$	$1.0 : 3$	≈ 0.66
Classificador C4.5				
SFS	19	$1min12s417ms$	$0.99\overline{653679} : 3$	≈ 0.663
SBS	0	$1min32s942ms$	$0.99\overline{69} : 7$	≈ 0.964
SFFS	19	$6min31s402ms$	$1.0 : 2$	≈ 0.372
BAB'	5	$2min34s941ms$	$0.99\overline{653679} : 7$	≈ 0.937
	10	$2min34s771ms$	$\approx 0.996 : 11$	≈ 0.893
	15	$24s316ms$	$0.99\overline{567099} : 18$	≈ 0.954
BAB	5	$5min612ms$	$0.99\overline{653679} : 7$	≈ 0.937
	10	$1min53s394ms$	$0.99\overline{567099} : 12$	≈ 0.915
	15	$37s491ms$	$0.99\overline{567099} : 17$	≈ 0.951
DEP	•	$2d16h54min27s611ms$	$1.0 : 2$	≈ 0.372

Tabela 4.18: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Image Segmentation*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $método = SBS^{C4.5}$						
SFS	≈ 0.67927	≈ 0.00949	≈ 71.57903	≈ 0.6884	≈ 0.77916	≈ 0.88351
SBS	≈ 0.90091	≈ 0.01218	≈ 73.96875	1	1	1
SFFS	≈ 0.67927	≈ 0.02257	≈ 30.09185	≈ 0.38592	≈ 4.21125	≈ 0.09164
DEP	≈ 0.68449	≈ 24.31874	≈ 0.02815	≈ 0.38592	≈ 2514.1229	≈ 0.00015
EB parcial $n \approx \frac{N}{4}$, $método = BAB'^{C4.5}$						
BAB'	≈ 0.94504	≈ 0.02039	≈ 46.34937	1	1	1
BAB	≈ 0.94504	≈ 2.26768	≈ 0.41674	1	≈ 1.97848	≈ 0.50544
EB parcial $n \approx \frac{N}{2}$, $método = BAB^{C4.5}$						
BAB'	≈ 0.99588	≈ 0.01074	≈ 92.71487	≈ 0.97665	≈ 1.3649	≈ 0.71555
BAB	≈ 0.99588	≈ 9.45776	≈ 0.1053	1	1	1
EB parcial $n \approx \frac{3 \cdot N}{4}$, $método = BAB'^{C4.5}$						
BAB'	≈ 0.93939	≈ 0.01641	≈ 57.24888	1	1	1
BAB	≈ 0.93939	≈ 1.2326	≈ 0.76212	≈ 0.99736	≈ 1.54182	≈ 0.64687

4.2.10 Parkinson

Contém 197 instâncias que armazenam medidas das vozes de 31 pessoas, 23 com doença de Parkinson, considerado como um problema binário. Cada padrões é formado por 22 características elas são: $MDVPFo$, $MDVPFhi$, $MDVPFlo$, $MDVPJitterPorc$, $MDVPJitterAbs$, $MDVPRAP$, $MDVPPPQ$, $JitterDDP$, $MDVPShimmer$, $MDVPShimmerdB$, $ShimmerAPQ3$, $ShimmerAPQ5$, $MDVPAPQ$, $ShimmerDDA$, NHR, HNR, $RPDE$, $D2$, DFA, $spread1$, $spread2$ e PPE[18], todas as características são discretas. Esta base de dados contém alguns valores que são muito pequenos, próximos a zero, essa base de dados pode indicar se o método implementado para Discretização de Características (DC, na Subseção 3.3.1, Página 48) é adequado para o uso com métodos para SC que utilizem Taxa de Inconsistência, além de ser uma das únicas duas bases de dados utilizadas que são classificadas como de médio porte por KUDO & SKLANSKY em [27]. A Tabela 4.19 mostra o resultado dos testes realizados.

A classificação com *holdout* para o conjunto com todas as características para a base de dados Parkinson resulta em uma precisão de ≈ 0.89. A Tabela 4.20 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Na Tabela 4.20, os métodos de destaque são os sequenciais (progressivo e regressivo) no primeiro grupo, e, os métodos BAB' com ambos critérios.

4.2.11 *Ionosphere*

Contém 351 padrões em duas classes: b (0, *bad* com 126 padrões) e g (1, *good* com 225 padrões). Os padrões representam medições recebidas por 16 antenas de alta frequência de um sistema de radar. Cada padrão é formado por 17 pulsos complexos, e existem duas características para cada pulso (coeficientes do número complexo), totalizando 34 características contínuas. Quando o sinal recebido é fraco (*bad*), significa que o sinal passa direto pela ionosfera, quando o sinal retornado é forte (*good*), significa que o sinal é refletido de volta por um objeto qualquer. O motivo pela escolha desta base de dados está relacionado à sua quantidade de características. Esta base de dados é uma das duas que são consideradas de médio porte por KUDO & SKLANSKY em [27], a outra é a Parkinson com 22 características. A Tabela 4.21

[18] Originalmente eram 23 características, uma característica era o identificador do indivíduo e foi removido.

Tabela 4.19: Testes sobre a base de dados Parkinson

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	22	$143ms$	$1.0 : 2$	≈ 0.82
SBS	0	$163ms$	$1.0 : 4$	≈ 0.89
SFFS	22	$330ms$	$1.0 : 2$	≈ 0.82
BAB'	5	$283ms$	$1.0 : 5$	≈ 0.85
	11	$86ms$	$1.0 : 11$	≈ 0.86
	17	$65ms$	$1.0 : 17$	≈ 0.89
BAB	5	$1min09s862ms$	$1.0 : 5$	≈ 0.85
	11	$9min50s652ms$	$1.0 : 11$	≈ 0.86
	17	$14s456ms$	$1.0 : 17$	≈ 0.89
DEP	•	$20min25s424ms$	$1.0 : 2$	≈ 0.79
Classificador C4.5				
SFS	22	$3s863ms$	$1.0 : 1$	≈ 0.70
SBS	0	$5s173ms$	$1.0 : 2$	≈ 0.85
SFFS	22	$10s508ms$	$1.0 : 1$	≈ 0.70
BAB'	5	$1min10s304ms$	$1.0 : 5$	≈ 0.89
	11	$7s740ms$	$1.0 : 11$	≈ 0.83
	17	$2s727ms$	$1.0 : 17$	≈ 0.90
BAB	5	$46min52s461ms$	$1.0 : 5$	≈ 0.87
	11	$5h13min56s683ms$	$1.0 : 11$	≈ 0.83
	17	$8min10s098ms$	$1.0 : 17$	≈ 0.90
DEP	•	$15h31min24s773ms$	$1.0 : 1$	≈ 0.70

mostra o resultado dos testes realizados[19].

A classificação com *holdout* para o conjunto com todas as características para a base de dados resulta em uma precisão de 0.8984375. A Tabela 4.22 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Na Tabela 4.22, dentre os métodos de ramificação e poda, no primeiro grupo ($n \approx \frac{N}{4}$) vê-se o ganho em desempenho obtido pelo algoritmo BAB' frente ao algotimo BAB, apesar do classificador C4.5 ser uma métrica muito mais custosa que a Taxa de Inconsistência, o método BAB'$^{C4.5}$ executou mais de 2000 vezes mais rapidamente que o método BAB^{CON} e encontrando uma solução ainda melhor.

[19]Devido ao tempo necessário para executar o método BAB^{CON} com $n = 8$, os testes utilizando os métodos DEP e os BAB restantes não foram realizados.

Tabela 4.20: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados Parkinson

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $método = SBS^{CON}$						
SFS	≈ 0.92063	≈ 0.8773	≈ 1.0494	≈ 0.79365	≈ 23.69939	≈ 0.03349
SBS	1	1	1	≈ 0.95238	≈ 31.7362	≈ 0.03001
SFFS	≈ 0.92063	≈ 2.02454	≈ 0.45474	≈ 0.79365	≈ 64.46626	≈ 0.01231
DEP	≈ 0.88889	≈ 7517.93865	≈ 0.00012	≈ 0.79365	≈ 342851.3681	≈ 0.0000023
EB parcial $n \approx \frac{N}{4}$, $método = BAB'^{C4.5}$						
BAB'	≈ 0.95238	≈ 0.00374	≈ 254.58628	1	1	1
BAB	≈ 0.95238	≈ 0.99371	≈ 0.95841	≈ 0.98413	≈ 40.00428	≈ 0.0246
EB parcial $n \approx \frac{N}{2}$, $método = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.96721	90	≈ 0.01075
BAB	1	6868.04651	0.00015	≈ 0.96721	≈ 219031.19767	≈ 0.0000044
EB parcial $n \approx \frac{3 \cdot N}{4}$, $método = BAB'^{C4.5}$						
BAB'	≈ 0.98438	≈ 0.02384	≈ 41.29832	1	1	1
BAB	≈ 0.98438	≈ 5.30106	≈ 0.18569	1	≈ 179.72057	≈ 0.00556

Tabela 4.21: Testes sobre a base de dados *Ionosphere*

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	34	$534ms$	$1.0 : 3$	0.875
SBS	0	$718ms$	$1.0 : 3$	0.828125
SFFS	34	$1s185ms$	$1.0 : 3$	0.875
BAB'	8	$536ms$	$1.0 : 8$	0.84375
	17	$340ms$	$1.0 : 17$	0.859375
	26	$146ms$	$1.0 : 26$	0.90625
BAB	8	$1d10hs04min30s793ms$	$1.0 : 8$	0.84375
Classificador C4.5				
SFS	34	$26s739ms$	$1.0 : 3$	0.75
SBS	0	$40s656ms$	$1.0 : 6$	0.875
SFFS	34	$1min33s471ms$	$1.0 : 3$	0.75
BAB'	8	$51s029ms$	$1.0 : 10$	0.875
	17	$5min23s774ms$	$1.0 : 18$	0.875
	26	$10s648ms$	$\approx 0.997 : 27$	0.8515625

Tabela 4.22: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Ionosphere*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método = SFS^{CON}}$						
SFS	1	1	1	≈ 0.85714	≈ 50.07303	≈ 0.01712
SBS	≈ 0.94643	≈ 1.34457	≈ 0.70389	1	≈ 76.13483	≈ 0.01313
SFFS	1	≈ 2.2191	≈ 0.45063	≈ 0.85714	≈ 175.03933	≈ 0.0049
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método = BAB'^{C4.5}}$						
BAB'	≈ 0.96429	≈ 0.0105	≈ 91.80324	1	1	1
BAB	≈ 0.96429	≈ 2403.94272	≈ 0.0004	$*$	$*$	$*$
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método = BAB'^{C4.5}}$						
BAB'	≈ 0.98214	≈ 0.00105	≈ 935.27153	1	1	1
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	≈ 0.93966	≈ 72.93151	≈ 0.01288

4.2.12 *Optical Recognition of Handwritten Digits*

Contém 5620 padrões em 10 classes (dígitos manuscritos), [0, 1, . . . , 9] com 554, 571, 557, 572, 568, 558, 558, 566, 554 e 562 padrões, respectivamente. Quarenta e três pessoas contribuíram com a escrita dos dígitos. As imagens dos cunhos dessas pessoas foram normalizadas em imagens binárias de 32x32 *pixels* que foram divididos em blocos de 4x4, a quantidade de *pixels* setados em cada bloco determina uma matriz de 8x8 (64 características) com valores inteiros entre [0, . . . , 16] que são os padrões contidos na base de dados. Esta base de dados é de grande porte e com quantidade de características aproximadamente uniforme entre as classes, das outras bases de dados de grande porte apenas a Madelon contém distribuição uniforme, contudo ela representa um problema binário e houve dificuldades na aplicação de alguns dos métodos. A Tabela 4.23 mostra o resultado dos testes realizados[20].

Tabela 4.23: Testes sobre a base de dados *Optical Recognition of Handwritten Digits*

Algoritmo	n	**Tempo**	$\Psi(S_\oplus) : \lvert S_\oplus \rvert$	$precisão(S_\oplus, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	64	$43s068ms$	$1.0 : 7$	≈ 0.7139
SBS	0	$1min10s238ms$	$1.0 : 9$	≈ 0.6132
SFFS	64	$1min58s353ms$	$1.0 : 7$	≈ 0.7139
BAB’	16	$2min04s856ms$	$1.0 : 16$	≈ 0.6223
	32	$23s818ms$	$1.0 : 32$	≈ 0.8279
	48	$8s835ms$	$1.0 : 48$	≈ 0.9014
Classificador C4.5				
SFS	64	$2h51min44s632ms$	$\approx 0.9957 : 9$	≈ 0.6516
SBS	0	$5h04min07s899ms$	$\approx 0.9936 : 25$	≈ 0.8961
SFFS	64	$1d18h26min52s608ms$	$\approx 0.998 : 11$	≈ 0.588
BAB’	16	$3hs11min23s196ms$	$\approx 0.9922 : 18$	≈ 0.83
	32	$15hs25min53s588ms$	$\approx 0.9916 : 41$	≈ 0.8855
	48	$2hs25min49s085ms$	$\approx 0.992 : 55$	≈ 0.889

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Optical Recognition of Handwritten Digits* resulta em uma precisão de ≈ 0.9057. A Tabela 4.24 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

[20]Levando em conta que os testes sobre métodos BAB e DEP não foram considerados viáveis na base de dados *Ionosphere* (Subseção 4.2.11 na Página 74) que tem 34 características e 351 padrões, os testes para este métodos (BAB e DEP) não foram realizados.

Sobre as taxas da Tabela 4.24, nota-se que o critério Taxa de Inconsistência na maioria das execuções não obteve bons resultados em proporção com os resultados obtidos pela aplicação do classificador C4.5, mesmo que utilizando resubstituição.

4.2.13 *Robot Execution Failures*

Contém 463 padrões que são medições dos valores de força e torque de um robô momentos antes a um estado de falha, no total são quinze medições em intervalos de tempo iguais das forças e torques em um espaço tridimensional (90 características). A base completa é formada por cinco diferentes problemas, cada qual está relacionado a uma dentre as seguintes tarefas:

- ao pegar (peça) com 88 padrões em quatro classes, $normal$ (0, sem falha), $collision$ (1, colisão), $frCollision$ (2, colisão frontal) e $obstruction$ (3, obstrução);
- ao transferir peça com 47 em cinco classes, $normal$ (0, sem falha), $frontCol$ (1, colisão frontal), $backCol$ (2, colisão traseira), $rightCol$ (3, colisão pela direita) e $leftCol$ (4, colisão pela esquerda);
- no posicionamento de uma peça antes de uma falha na transferência com 47 padrões em quatro classes, ok (0, sem problemas), $slightlyMoved$ (1, ligeiramente movido), $moved$ (2, movido) e $lost$ (3, perdido);
- ao soltar (peça) 117 padrões em três classes, $normal$ (0, sem falha), $collision$ (1, colisão) e $obstruction$ (2, obstrução);
- ao movimentar peça com 164 padrões em cinco classes, $normal$ (0, sem falhas), $bottomCollision$ (1, colisão no fundo), $bottomObstruction$ (2, obstrução no fundo), $collisionInPart$ (3, colisão na peça) e $collisionInTool$ (4, colisão na ferramenta, atuador).

A escolha destas bases é pela relação entre quantidade de características (N muito pequeno) e quantidade de padrões ($M = 90$), o que pode ocasionar problemas de classificação devido ao Problema da Dimensionalidade (PD) descrito no Capítulo 2, na Página 3. As Tabelas 4.25, 4.27, 4.29, 4.31 e 4.33 mostram os resultados dos testes realizados nas cinco bases de dados[21].

[21] Os métodos BAB e DEP não foram executados por não terem sido considerados viáveis em outras bases de dados, o que acarreta viabilidade para esta base de dados devido à baixa quantidade de padrões.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Robot Execution Failures* ao pegar (peça) resulta em uma precisão de 0.78125. A Tabela 4.26 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Pela Tabela 4.26, observa-se que em todas as execuções ambos métodos com quaisquer dos critérios obtiveram soluções equivalentes, sendo que as versões que utilizam a Taxa de Inconsitência levaram menos tempo para executar do que as versões com critério C4.5.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Robot Execution Failures* ao transferir peça resulta em uma precisão de $0.4\overline{6}$. A Tabela 4.28 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Na Tabela 4.28 a utilização dos dois critérios resultou em subconjuntos de características equivalentes para cada algoritmo, a Taxa de Inconsitência é superior com relação ao tempo necessário de processamento o que conta a favor com se une os dois quesitos pela observação da coluna que indica o custo benefício do processamento (av).

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Robot Execution Failures* no posicionamento de uma peça antes de uma falha na transferência resulta em uma precisão de $0.7\overline{3}$. A Tabela 4.30 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

A Tabela 4.30 indica que ambos os critérios são equivalentes com relação a qualidade da solução encontrada, porém, a Taxa de Inconsistência é menos custosa. O algoritmo SBS não foi bem sucedido ao buscar a solução possívelmente por remover uma característica que seja essencial ao processo classificatório.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Robot Execution Failures* ao soltar (peça) resulta em uma precisão de ≈ 0.88. A Tabela 4.32 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Na Tabela 4.32 os métodos com critério monotônico foram superiores, exceto pelo grupo de ramificação e poda com $n \approx \frac{N}{4}$ onde a versão que usa o classificador C4.5 achou solução superior ao custo de maior tempo de processamento.

A classificação com *holdout* para o conjunto com todas as características para a base de dados *Robot Execution Failures* ao movimentar peça resulta em uma precisão de ≈ 0.57. A Tabela 4.34 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo.

Pela Tabela 4.34 prevalesce o uso do critério monotônico sobre o classificador C4.5 como métrica, apesar de encontrar melhores soluções em dois casos nos grupos que exploram o EB de forma parcial quando $n \approx \frac{N}{4}$ e $n \approx \frac{3 \cdot N}{4}$.

Tabela 4.24: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Optical Recognition of Handwritten Digits*

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método} = SBS^{C4.5}$						
SFS	≈ 0.79667	≈ 0.00578	≈ 137.77102	≈ 0.72711	≈ 1.38356	≈ 0.52554
SBS	≈ 0.6843	≈ 0.00943	≈ 72.56229	1	1	1
SFFS	≈ 0.79667	≈ 0.01589	≈ 50.13411	≈ 0.65577	≈ 20.51755	≈ 0.03196
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método} = BAB'^{C4.5}$						
BAB'	≈ 0.74968	≈ 0.01087	≈ 68.94912	1	1	1
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método} = BAB'^{C4.5}$						
BAB'	≈ 0.93502	≈ 0.00043	≈ 2180.85513	1	1	1
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	≈ 0.98582	≈ 990.27561	≈ 0.001

Tabela 4.25: Testes sobre a base de dados *Robot Execution Failures* ao pegar (peça)

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	90	$1s411ms$	$1.0 : 2$	0.59375
SBS	0	$2s173ms$	$1.0 : 3$	0.5
SFFS	90	$3s588ms$	$1.0 : 2$	0.59375
BAB'	22	$1s484ms$	$1.0 : 22$	0.65625
	45	$822ms$	$1.0 : 45$	0.53125
	68	$355ms$	$1.0 : 68$	0.625
Classificador C4.5				
SFS	90	$1min30s948ms$	$1.0 : 2$	0.59375
SBS	0	$2min51s092ms$	$1.0 : 3$	0.5
SFFS	90	$4min21s054ms$	$1.0 : 2$	0.59375
BAB'	22	$1min46s175ms$	$1.0 : 22$	0.65625
	45	$5min09s417ms$	$1.0 : 45$	0.53125
	68	$10min36s646ms$	$1.0 : 68$	0.625

Tabela 4.26: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Robot Execution Failures* ao pegar (peça)

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método = SFS^{CON}}$						
SFS	1	1	1	1	≈ 64.45641	≈ 0.01551
SBS	≈ 0.84211	≈ 1.54004	≈ 0.54681	≈ 0.84211	≈ 121.25585	≈ 0.00694
SFFS	1	≈ 2.54288	≈ 0.39326	1	≈ 185.01347	≈ 0.00541
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 71.5465	≈ 0.01398
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 376.41971	≈ 0.00266
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 1793.36901	≈ 0.00056

Tabela 4.27: Testes sobre a base de dados *Robot Execution Failures* ao transferir peça

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	90	$856ms$	$1.0 : 2$	0.6
SBS	0	$1s268ms$	$1.0 : 3$	$0.2\overline{6}$
SFFS	90	$2s018ms$	$1.0 : 2$	0.6
BAB'	22	$907ms$	$1.0 : 22$	0.4
	45	$521ms$	$1.0 : 45$	$0.\overline{3}$
	68	$225ms$	$1.0 : 68$	$0.5\overline{3}$
Classificador C4.5				
SFS	90	$56s589ms$	$1.0 : 2$	0.6
SBS	0	$1min47s589ms$	$1.0 : 3$	$0.2\overline{6}$
SFFS	90	$2min46s910ms$	$1.0 : 2$	0.6
BAB'	22	$1min06s973ms$	$1.0 : 22$	0.4
	45	$35s113ms$	$1.0 : 45$	$0.\overline{3}$
	68	$12s129ms$	$1.0 : 68$	$0.5\overline{3}$

Tabela 4.28: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Robot Execution Failures* ao transferir peça

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método = SFS^{CON}}$						
SFS	1	1	1	1	≈ 66.10864	≈ 0.01513
SBS	≈ 0.44444	≈ 1.48131	≈ 0.30004	≈ 0.44444	≈ 125.68808	≈ 0.00354
SFFS	1	≈ 2.35748	≈ 0.42418	1	≈ 194.98832	≈ 0.00513
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 73.84013	≈ 0.01354
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 67.39539	≈ 0.01484
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 53.90667	≈ 0.01855

Tabela 4.29: Testes sobre a base de dados *Robot Execution Failures* no posicionamento de uma peça antes de uma falha na transferência

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	90	$857ms$	$1.0 : 2$	$0.7\overline{3}$
SBS	0	$1s272ms$	$1.0 : 3$	0.4
SFFS	90	$2s020ms$	$1.0 : 2$	$0.7\overline{3}$
BAB'	22	$906ms$	$1.0 : 22$	0.6
	45	$517ms$	$1.0 : 45$	$0.7\overline{3}$
	68	$224ms$	$1.0 : 68$	0.6
Classificador C4.5				
SFS	90	$49s887ms$	$1.0 : 1$	$0.7\overline{3}$
SBS	0	$1min28s990ms$	$1.0 : 3$	0.4
SFFS	90	$2min26s381ms$	$1.0 : 2$	$0.7\overline{3}$
BAB'	22	$58s965ms$	$1.0 : 22$	0.6
	45	$30s749ms$	$1.0 : 45$	$0.7\overline{3}$
	68	$10s772ms$	$1.0 : 68$	0.6

Tabela 4.30: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Robot Execution Failures* no posicionamento de uma peça antes de uma falha na transferência

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método} = SFS^{CON}$						
SFS	1	1	1	1	≈ 58.2112	≈ 0.01718
SBS	≈ 0.54545	≈ 1.48425	≈ 0.3675	≈ 0.54545	≈ 103.83897	≈ 0.00525
SFFS	1	≈ 2.35706	≈ 0.42426	1	≈ 170.8063	≈ 0.00585
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	1	≈ 65.08278	≈ 0.01537
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	1	≈ 59.47582	≈ 0.01681
EB parcial $n \approx \frac{3\cdot N}{4}$, $\widetilde{método} = BAB'^{CON}$						
BAB'	1	1	1	1	≈ 48.08929	≈ 0.02079

Tabela 4.31: Testes sobre a base de dados *Robot Execution Failures* ao soltar (peça)

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	90	$1s838ms$	$1.0 : 2$	≈ 0.83
SBS	0	$2s924ms$	$1.0 : 3$	$0.\overline{714285}$
SFFS	90	$4s788ms$	$1.0 : 2$	≈ 0.83
BAB'	22	$1s993ms$	$1.0 : 22$	$0.\overline{7380952}$
	45	$1s090ms$	$1.0 : 45$	≈ 0.88
	68	$452ms$	$1.0 : 68$	≈ 0.83
Classificador C4.5				
SFS	90	$1min40s653ms$	$1.0 : 2$	$0.\overline{809523}$
SBS	0	$2min46s149ms$	$1.0 : 3$	$0.\overline{714285}$
SFFS	90	$4min44s323ms$	$1.0 : 2$	$0.\overline{809523}$
BAB'	22	$27min45s805ms$	$1.0 : 22$	≈ 0.83
	45	$10min38s049ms$	$1.0 : 45$	≈ 0.83
	68	$1min54s041ms$	$1.0 : 68$	≈ 0.83

Tabela 4.32: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Robot Execution Failures* ao soltar (peça)

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método = SFS^{CON}}$						
SFS	1	1	1	≈ 0.97143	≈ 54.76224	≈ 0.01774
SBS	≈ 0.85714	≈ 1.59086	≈ 0.53879	≈ 0.85714	≈ 90.39663	≈ 0.00948
SFFS	1	≈ 2.60501	≈ 0.38388	≈ 0.97143	≈ 154.69151	≈ 0.00628
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método = BAB'^{C4.5}}$						
BAB'	≈ 0.88571	≈ 0.0012	≈ 740.30471	1	1	1
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	≈ 0.94595	≈ 585.36606	≈ 0.00162
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 252.3031	≈ 0.00396

Tabela 4.33: Testes sobre a base de dados *Robot Execution Failures* ao movimentar peça

Algoritmo	n	**Tempo**	$\Psi(S_{\oplus}) : \lvert S_{\oplus}\rvert$	$precisão(S_{\oplus}, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	90	$2s544ms$	$\approx 0.99 : 2$	≈ 0.57
SBS	0	$4s381ms$	$\approx 0.99 : 3$	≈ 0.51
SFFS	90	$6s966ms$	$\approx 0.99 : 2$	≈ 0.57
BAB'	22	$2s909ms$	$\approx 0.99 : 22$	≈ 0.56
	45	$1s572ms$	$\approx 0.99 : 45$	≈ 0.56
	68	$624ms$	$\approx 0.99 : 68$	≈ 0.59
Classificador C4.5				
SFS	90	$4min22s005ms$	$\approx 0.99 : 2$	≈ 0.51
SBS	0	$7min52s374ms$	$\approx 0.99 : 3$	≈ 0.51
SFFS	90	$12min44s129ms$	$\approx 0.99 : 2$	≈ 0.51
BAB'	22	$14min24s906ms$	$\approx 0.99 : 22$	≈ 0.59
	45	$2min33s703ms$	$\approx 0.99 : 45$	≈ 0.56
	68	$50s630ms$	$\approx 0.99 : 68$	≈ 0.62

Tabela 4.34: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados *Robot Execution Failures* ao movimentar peça

Critério	Taxa de Inconsistência			Classificador C4.5		
Algoritmo	**ac**	**ms**	**av**	**ac**	**ms**	**av**
Todo o EB, $\widetilde{método = SFS^{CON}}$						
SFS	1	1	1	≈ 0.88571	≈ 102.98939	≈ 0.0086
SBS	≈ 0.88571	≈ 1.72209	≈ 0.51432	≈ 0.88571	≈ 185.6816	≈ 0.00477
SFFS	1	≈ 2.73821	≈ 0.3652	≈ 0.88571	≈ 300.36517	≈ 0.00295
EB parcial $n \approx \frac{N}{4}$, $\widetilde{método = BAB'^{C4.5}}$						
BAB'	≈ 0.94444	≈ 0.00336	≈ 280.80291	1	1	1
EB parcial $n \approx \frac{N}{2}$, $\widetilde{método = BAB'^{CON}}$						
BAB'	1	1	1	1	≈ 97.77545	≈ 0.01023
EB parcial $n \approx \frac{3 \cdot N}{4}$, $\widetilde{método = BAB'^{C4.5}}$						
BAB'	≈ 0.94737	≈ 0.01232	≈ 76.86741	1	1	1

4.2.14 Madelon

Contém 2600 padrões em duas classes 1 e 0[22], com quantidades iguais de padrões, cada padrão é formado por 500 características[23]. É uma base de dados artificial criada para o Desafio NIPS 2003 de Seleção de Características[24], GUYON em [15] e GUYON *et al.* em [16]. A Tabela 4.35 mostra o resultado dos testes realizados[25].

Tabela 4.35: Testes sobre a base de dados Madelon

Algoritmo	n	**Tempo**	$\Psi(S_\oplus) : \lvert S_\oplus \rvert$	$precisão(S_\oplus, \frac{2\cdot\Upsilon}{3}, \frac{\Upsilon}{3})$
Taxa de Inconsistência				
SFS	500	$2h10min44s332ms$	$1.0 : 3$	≈ 0.55
SBS	0	$4h25min08s838ms$	$1.0 : 4$	≈ 0.50
SFFS	500	$6h38min34s585ms$	$1.0 : 3$	≈ 0.55
BAB'	125	$2h51min40s451ms$	$1.0 : 125$	≈ 0.74
	250	$1h26min06s411ms$	$1.0 : 250$	≈ 0.73
	375	$25min14s698ms$	$1.0 : 375$	≈ 0.73

A classificação com *holdout* para o conjunto com todas as características para a base de dados Madelon resulta em uma precisão de ≈ 0.72. A Tabela 4.36 contém os valores calculados para as taxas de ganho em precisão, em tempo e em precisão por tempo[26].

Tabela 4.36: Taxas de ganho em precisão, tempo e precisão por tempo computadas para a base de dados Madelon

Critério	Taxa de Inconsistência		
Algoritmo	**ac**	**ms**	**av**
Todo o EB, $método = SFS^{CON}$			
SFS	1	1	1
SBS	≈ 0.90619	≈ 2.02807	≈ 0.44682
SFFS	1	≈ 3.04865	≈ 0.32801

Na Tabela 4.36 o melhor método, dentre os comparados, foi o Busca Sequencial Progressiva

[22] A classe 0 era originalmente definida como -1. Devido a um pré-requisito da implementação utilizada para o classificador C4.5 essa classe foi renomeada para 0.

[23] Originalmente, continha 4400 padrões (2000 para treinamento, 600 para validação e 1800 para teste), mas como o repositório não distribui o arquivo que contém as classes dos 1800 padrões para teste, apenas 2600 foram utilizados.

[24] *NIPS 2003 feature selection challenge*. NIPS é a sigla para *Neural Information Processing Systems Conference*.

[25] Tentou-se executar o método $SFS^{C4.5}$ durante cinco dias sem êxito. Dessa forma, os métodos que usam como critério o classificador C4.5 foram considerados inviáveis.

[26] As taxas referentes aos grupos que exploram o EB de forma parcial não foram computados porque apenas um método pôde ser aplicado.

utilizando Taxa de Inconsistência como critério, os outros dois métodos (SBS e SFFS) levaram o dobro e o triplo de tempo, respectivamente, para terminar a execução.

4.3 Comparação dos Métodos

Nesta seção os doze métodos descritos no Capítulo 3 são comparados com relação ao seus custos benefícios (Equação 4.3) determinados pelos testes apresentados durante a explanação das bases de dados utilizadas. A comparação é realizada pela aplicação da Equação 4.7 que, no contexto desta monografia, dada as médias e os desvios padrões determinados por uma série de execuções idênticas de dois métodos distintos (A e B), a referida equação retorna a diferença absoluta entre os dois métodos em termos de desvios padrões. Caso $||método_A - método_B|| \leqslant 0$, então o método B é superior ao método A, caso contrário o método A é superior ao B. Se a diferença absoluta entre os dois métodos for superior (ou menor ou igual) a dois desvios padrões, a afirmação é válida com grau de confiança de 95%[27]. Durante a comparação dos métodos nos casos em que a diferença absoluta não superar a unidade, os métodos serão considerados estatisticamente equivalentes.

A Tabela 4.37 relaciona os métodos de cada grupo comparativo com suas médias e desvios padrões para cada subconjunto amostral, quantidades de testes válidos. Devido à inviabilidade da aplicação de alguns dos métodos sobre algumas das bases de dados, devido à natureza exponencial do problema, e para aproveitar ao máximo todas as execuções dos métodos, quando se comparar dois métodos onde uma ou mais execuções de um dos métodos não pôde ser concretizada, a(s) execução(ões) simétrica(s) do outro método são desconsideradas, ou seja, durante o confronto entre dois métodos considera-se todas as execuções concretizadas para ambos os métodos.

Nas Tabelas 4.38, 4.39, 4.40 e 4.41 estão representados os cômputos da Equação 4.7 para confrontar todos os métodos uns com os outros dentro de cada grupo comparativo. Nessas três tabelas as células devem ser interpretadas como o resultado de $||método_{linha} - método_{coluna}||$, calculado utilizando as médias e desvios padrões obtidos da Tabela 4.37 com o maior valor de q (número após os dois pontos) para qual existem entradas para ambos os métodos.

[27]Caso a diferença seja maior que um desvio padrão a afirmação é válida com grau de confiança de 68%. De forma idêntica, se a diferença for de ordem superior a três unidades, a afirmação é válida com 99.7% de confiabilidade.

Pela Tabela 4.38, pode-se dizer que os métodos SFS e SBS, independentemente do critério utilizado, são equivalentes com relação ao custo benefício. De um modo geral o uso da Taxa de Inconsistência como critério é uma melhor opção do que o classificador C4.5, com resubstituição pelo menos. O método que melhor se adequa na comparação com todos os outros métodos comparados que exploram todo o espaço de busca é o SFFS^{CON}, pois o tempo excedente necessário para sua execução é compensado pela obtenção de melhores soluções. A Figura 4.1 contém um gráfico de barras com os valores, da Tabela 4.38, aculados para cada método.

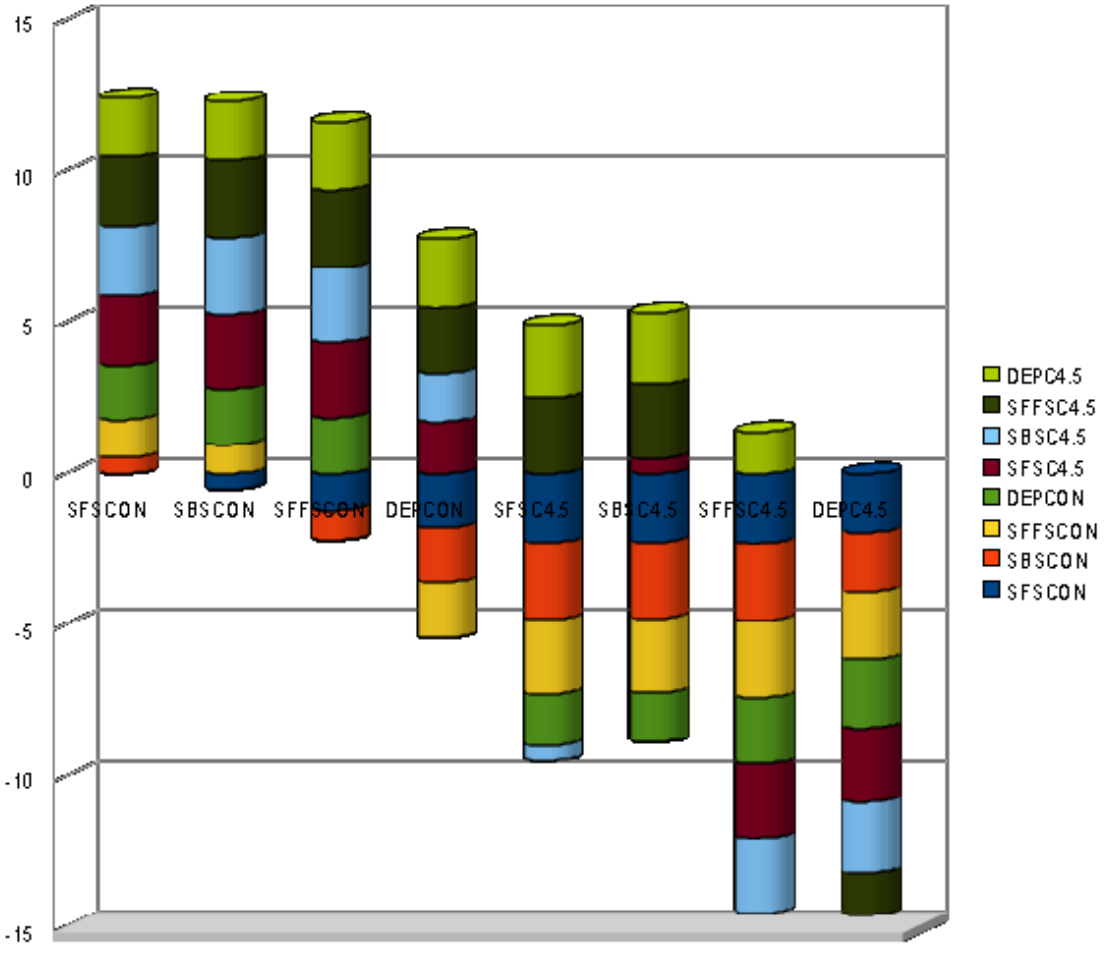


Figura 4.1: Gráfico de barras acumulativas dos confrontos entre os métodos que exploram todo o EB

Na Figura 4.1 cada cilindro representa a soma dos valores obtidos pelos confrontos entre os métodos que exploram todo o EB, da esquerda para a direita tem-se os métodos SFS$^{(}CON$), SBS$^{(}CON$), SFFS$^{(}CON$), DEP$^{(}CON$), SFS$^{(}C4.5$), SBS$^{(}C4.5$), SFFS$^{(}C4.5$) e DEP$^{(}C4.5$).

Esta figura propicia uma comprovação visual dos resultados declarados anteriormente a partir da Tabela 4.38. Métodos que utilizam os algoritmos SFS e SBS são equivalentes independentemente do critério utilizado, SFS$^{(}CON)$ equivale ao SBS$^{(}CON)$ e SFS$^{(}C4.5)$ equivale ao SBS$^{(}C4.5)$, apesar de necessitar de um maior tempo de processamento o métodos SFFS^{CON} compensa pela qualidade da solução que encontra. Todos os métodos que utilizam o critério Taxa de Inconsistência foram superioes a qualquer método com o critério C4.5 utilizando resubstituição. Ressalta-se que os métodos exaustivos não puderam ser aplicados a todas as bases de dados, portanto, os valores obtidos para tais métodos estão super estimados, caso fosse possível suas execuções o esperado é que os valores de custo pelo benefício fossem drasticamente reduzidos.

Dentre os métodos que exploram o EB de forma parcial de acordo com a Tabela 4.39 ($n \approx \frac{N}{4}$), a Taxa de Inconsitência também mostrou-se mais adequada que o classificador C4.5 com resubstituição. O método melhor classificado de acordo com o custo benefício da computação é o BAB'^{CON}, nota-se que o método BAB^{CON} também obteve bons resultados, porém, no confronto direto entre os dois métodos com critério de consistência a versão alterada (BAB'^{CON}) vence a versão original por uma diferença superior a dois desvios padrões. A Figura 4.2 contém um gráfico de barras com os valores, da Tabela 4.39, aculados para cada método.

Na Figura 4.2 cada cilindro representa a soma dos valores obtidos pelos confrontos entre os métodos que exploram três quartos do EB ($n \approx \frac{(}{N})4$), da esquerda para a direita tem-se os métodos BAB'^{CON}, BAB^{CON}, BAB'$^{C4.5}$ e BAB$^{C4.5}$. Esta figura propicia uma comprovação visual dos resultados declarados anteriormente a partir da Tabela 4.39. Os métodos com critério Taxa de Inconsistência foram superiores e apesar do método BAB^{CON} estar muito bem colocado, ele perde no confronto direto com o método BAB'^{CON}.

Observando a Tabela 4.40 que considera a computação para metade do EB em comparação com a Tabela 4.39, nota-se que a eficiência do método BAB'^{CON} caiu ainda mais frente ao método BAB^{CON} está inferior a dois desvios padrões, entretanto, a Taxa de Inconsitência continua superior ao classificador C4.5. A Figura 4.3 contém um gráfico de barras com os valores, da Tabela 4.40, aculados para cada método.

Na Figura 4.3 cada cilindro representa a soma dos valores obtidos pelos confrontos entre

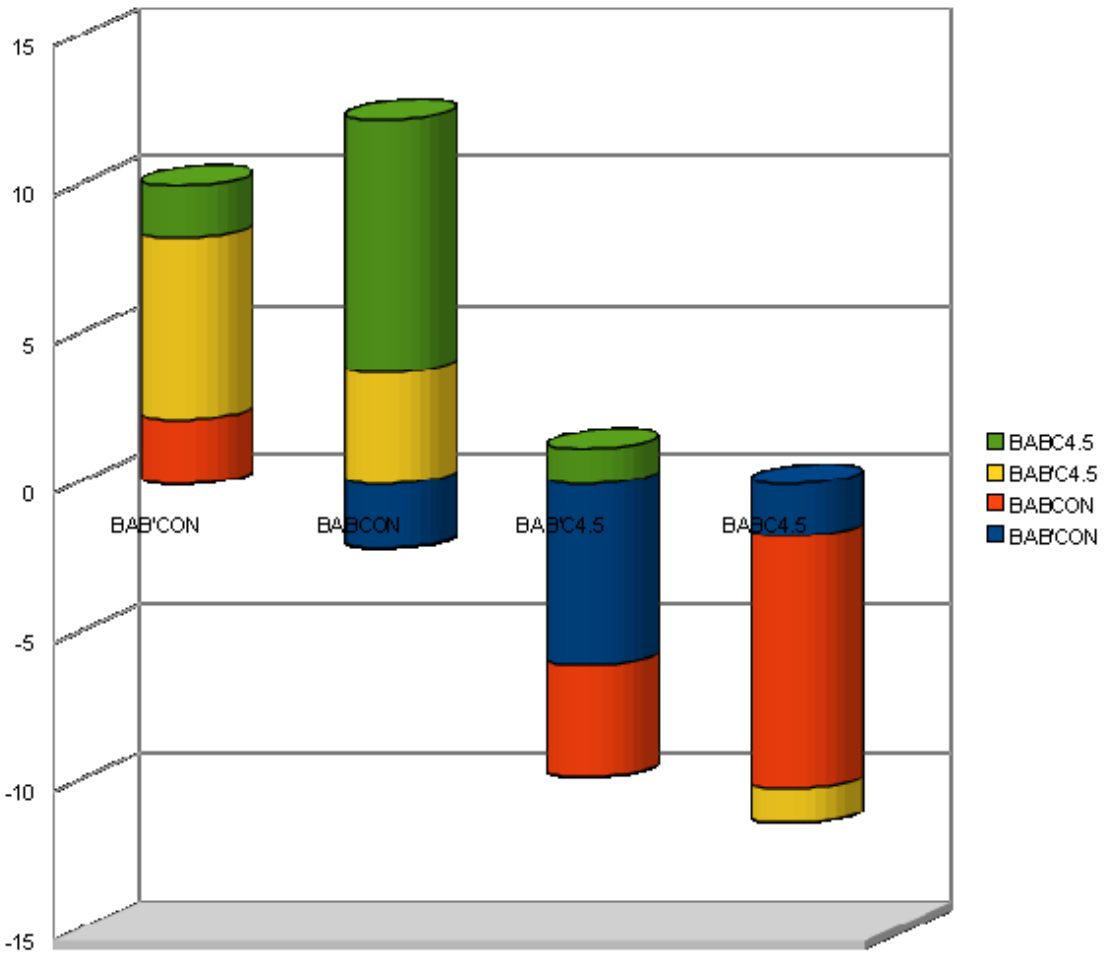


Figura 4.2: Gráfico de barras acumulativas dos confrontos entre os métodos que exploram três quartos do EB

os métodos que exploram metade do EB ($n \approx \frac{(}{N})2$), da esquerda para a direita tem-se os métodos BAB'^{CON}, BAB^{CON}, BAB'$^{C4.5}$ e BAB$^{C4.5}$. Esta figura propicia uma comprovação visual dos resultados declarados anteriormente a partir da Tabela 4.40. Os métodos com critério Taxa de Inconsistência continuam superiores, a diferença entre os resultados para este grupo comparativo ($n \approx \frac{N}{2}$) frente ao anterior ($n \approx \frac{N}{4}$) reside na qualificação obtida pelo método BAB^{CON} frente ao método BAB'^{CON} que ficou muito mais visível. O método BAB^{CON} obteve o melhor comportamento frente as bases de dados testadas.

De acordo com a comparação representada pela Tabela 4.41 para $n \approx \frac{3 \cdot N}{4}$, o melhor método dentre os considerados neste grupo comparativo é o método BAB'^{CON}, como pode ser visualmente comprovado pela Figura 4.4. Esta figura contém um gráfico de barras com os valores, da

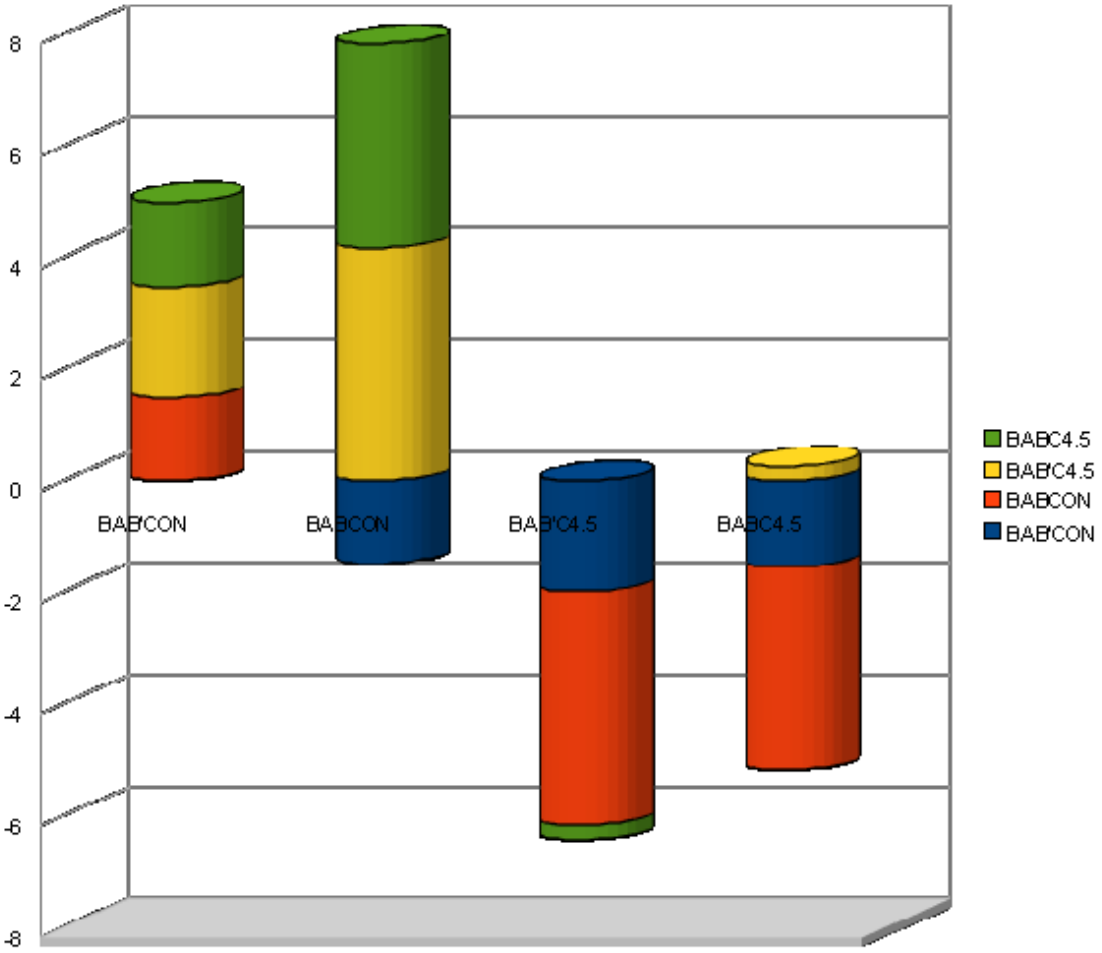


Figura 4.3: Gráfico de barras acumulativas dos confrontos entre os métodos que exploram metade do EB

Tabela 4.41, aculados para cada método.

Na Figura 4.4 cada cilindro representa a soma dos valores obtidos pelos confrontos entre os métodos que exploram um quarto do EB ($n \approx \frac{(}{3} \cdot N)4$), da esquerda para a direita tem-se os métodos BAB'^{CON}, BAB^{CON}, BAB'$^{C4.5}$ e BAB$^{C4.5}$. Esta figura propicia uma comprovação visual dos resultados declarados anteriormente a partir da Tabela 4.41. Os métodos com critério Taxa de Inconsistência continuam superiores. Neste grupo comparativo a avaliação do método BAB'^{CON} é muito superior ao método BAB^{CON}.

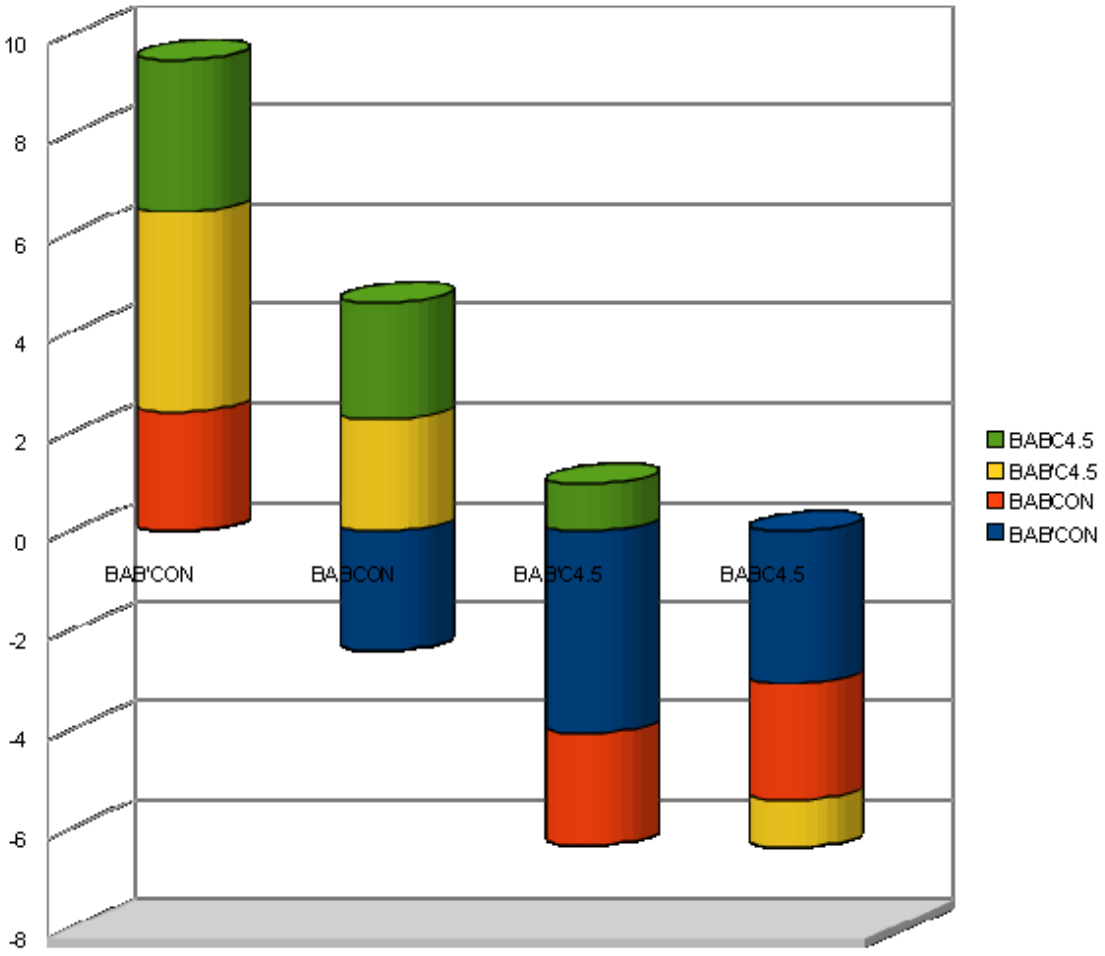


Figura 4.4: Gráfico de barras acumulativas dos confrontos entre os métodos que exploram um quarto do EB

Tabela 4.37: Médias e desvios padrões dos métodos de acordo com os grupos comparativos

Critério		Taxa de Inconsistência		Classificador C4.5	
Algoritmo	q	$\overline{av}(método)$	$\sigma(método)$	$\overline{av}(método)$	$\sigma(método)$
Todo o EB					
SFS	9	10.8125056	7.8330664	0.2428356	0.1185264
SBS	9	11.2742911	8.118692	0.2684944	0.1388148
SFFS	9	5.4629633	3.4491281	0.0692878	0.0299809
DEP	9	0.9456211	0.5672644	0.0359147	0.0164301
SFS	10	9.865183	7.0698631	0.219039	0.1086512
SBS	10	10.246862	7.3339034	0.241943	0.126967
SFFS	10	4.951349	3.1271292	0.062455	0.0276726
DEP	10	0.851293	0.5160706	$*$	$*$
SFS	17	14.2601676	8.7131334	0.16513	0.0694012
SBS	17	10.4707212	5.7828376	0.2036788	0.0910331
SFFS	17	6.0052359	3.3006305	0.0404135	0.0170181
SFS	18	13.5234917	8.2553086	$*$	$*$
SBS	18	9.9138378	5.4863776	$*$	$*$
SFFS	18	5.6898344	3.1311011	$*$	$*$
EB parcial $n \approx \frac{N}{4}$					
BAB'	10	31.335453	25.2060167	0.340586	0.1445402
BAB	10	0.977042	0.0818661	0.195207	0.1014286
BAB'	11	36.8325245	23.4530193	0.4005327	0.1438296
BAB	11	0.8882564	0.1156131	$*$	$*$
BAB'	17	88.1302653	20.3457123	0.4381618	0.1075426
EB parcial $n \approx \frac{N}{2}$					
BAB'	10	10.167009	9.1719856	0.144919	0.0661524
BAB	10	0.596576	0.137659	0.1661474	0.0946933
BAB'	17	189.5762794	135.0025061	0.2056088	0.081485
EB parcial $n \approx \frac{3 \cdot N}{4}$					
BAB'	10	19.619156	8.9958304	0.442015	0.1518255
BAB	10	3.904315	2.1588952	0.307301	0.1293804
BAB'	17	16.4152335	5.6144657	0.3222288	0.0996965

Tabela 4.38: Comparação dois a dois entre os métodos que exploram todo o EB

Métodos	**SFS**^{CON}	**SBS**^{CON}	**SFFS**^{CON}	**DEP**^{CON}	**SFS**$^{C4.5}$	**SBS**$^{C4.5}$	**SFFS**$^{C4.5}$	**DEP**$^{C4.5}$
SFS^{CON}	$0:18$	$0,52:18$	$1,25:18$	$1,8:10$	$2,29:17$	$2,28:17$	$2,31:17$	$1,95:9$
SBS^{CON}	$-0,52:18$	$0:18$	$0,95:18$	$1,81:10$	$2,52:17$	$2,51:17$	$2,55:17$	$1,96:9$
SFFS^{CON}	$-1,25:18$	$-0,95:18$	$0:18$	$1,83:10$	$2,5:17$	$2,48:17$	$2,56:17$	$2,23:9$
DEP^{CON}	$-1,8:10$	$-1,81:10$	$-1,83:10$	$0:10$	$1,7:10$	$1,62:10$	$2,16:10$	$2,27:9$
SFS$^{C4.5}$	$-2,29:17$	$-2,52:17$	$-2,5:17$	$-1,7:10$	$0:17$	$-0,48:17$	$2,47:17$	$2,45:9$
SBS$^{C4.5}$	$-2,28:17$	$-2,51:17$	$-2,48:17$	$-1,62:10$	$0,48:17$	$0:17$	$2,49:17$	$2,35:9$
SFFS$^{C4.5}$	$-2,31:17$	$-2,55:17$	$-2,56:17$	$-2,16:10$	$-2,47:17$	$-2,49:17$	$0:17$	$1,38:9$
DEP$^{C4.5}$	$-1,95:9$	$-1,96:9$	$-2,23:9$	$-2,27:9$	$-2,45:9$	$-2,35:9$	$-1,38:9$	0

Tabela 4.39: Comparação dois a dois entre os métodos de ramificação e poda com $n \approx \frac{N}{4}$

Métodos	**BAB'**^{CON}	**BAB**^{CON}	**BAB'**$^{C4.5}$	**BAB**$^{C4.5}$
BAB'^{CON}	$0:17$	$2,17:11$	$6,1::17$	$1,75::10$
BAB^{CON}	$-2,17:11$	$0:11$	$3,74:11$	$8,48:10$
BAB'$^{C4.5}$	$-6,1:17$	$-3,74:11$	$0:17$	$1,16:10$
BAB$^{C4.5}$	$-1,75:10$	$-8,48:10$	$-1,16:10$	$0:10$

Tabela 4.40: Comparação dois a dois entre os métodos de ramificação e poda com $n \approx \frac{N}{2}$

Métodos	**BAB'**^{CON}	**BAB**^{CON}	**BAB'**$^{C4.5}$	**BAB**$^{C4.5}$
BAB'^{CON}	$0:17$	$1,48:10$	$1,98:17$	$1,54:10$
BAB^{CON}	$-1,48:10$	$0:10$	$4,18:10$	$3,64:10$
BAB'$^{C4.5}$	$-1,98:17$	$-4,18:10$	$0:17$	$-0,26:10$
BAB$^{C4.5}$	$-1,54:10$	$-3,64:10$	$0,26:10$	$0:10$

Tabela 4.41: Comparação dois a dois entre os métodos de ramificação e poda com $n \approx \frac{3 \cdot N}{4}$

Métodos	**BAB'**^{CON}	**BAB**^{CON}	**BAB'**$^{C4.5}$	**BAB**$^{C4.5}$
BAB'^{CON}	$0:17$	$2,4:10$	$4,05:17$	$3,04:10$
BAB^{CON}	$-2,4:10$	$0:10$	$2,26:10$	$2,35:10$
BAB'$^{C4.5}$	$-4,05:17$	$-2,26:10$	$0:17$	$0,96:10$
BAB$^{C4.5}$	$-3,04:10$	$-2,35:10$	$-0,96:10$	$0:10$

Capítulo 5

Conclusão

Nesta monografia realizou-se a comparação entre doze métodos para Seleção de Características. Os doze métodos foram obtidos pela combinação de seis algoritmos (SFS, SBS, SFFS, BAB', BAB e DEP) com duas funções critérios (Taxa de Inconsistência e classificador C4.5 com abordagem resubstitutiva). Os doze métodos foram divididos em quatro grupos comparativos conforme a quantidade do Espaço de Busca em que eles atuam, todo o Espaço de busca ou parcial. A exploração parcial do Espaço de Busca se ramifica em três grupos comparativos, um quarto, metade e três quartos do Espaço de Busca.

O critério para realizar a comparação entre os métodos em cada grupo foi a proporção entre a qualidade dos subconjuntos obtidos sobre o tempo de processamento necessário, o custo benefício da aplicação dos métodos para Seleção de Características. Essa forma de comparação foi escolhida devido a sua aplicabilidade prática, para problemas onde o processo para Seleção de Atributos, dependendo do método aplicado, não seja viável e onde não apenas a qualidade da solução é um fator importante, mas também, o tempo que se dispõe para tal.

Para realizar a comparação, foram utilizadas dezoito bases de testes. A escolha das bases de testes foi realizada com base em uma série de critérios: quantidades de características, padrões, classes, proporção de padrões por classes desde uniformes até casos onde uma ou mais classes eram privilegiadas, qualidades dos atributos (relevantes, redundantes e irreleventes). Após a execução dos testes, independentemente do critério utilizado, as melhores soluções encotradas foram reavaliadas pelo C4.5, induziu-se um novo classificador com a abordagem *holdout*, com proporções de treinamento e teste iguais a $\frac{1}{3}$ e $\frac{2}{3}$ respectivamente.

Os resultados obtidos são que de uma forma geral métodos que utilizam como critério a Taxa de Inconsitência são mais eficientes que métodos que se valem do classificador C4.5 como

métrica, pelo menos com o uso da metodologia da resubstituição para determinar a precisão. O fato de calcular a precisão com uma abordagem e avaliar posteriormente com outra afetou a qualidade das soluções dos métodos com este critério.

Dentro do grupo comparativo que explora todo o Espaço de Busca, o mais indicado é o algoritmo Busca Sequencial Flutuante Progressiva aliado ao critério Taxa de Inconsitência, caso o tempo disponível seja um fator crítico os outros dois métodos sequencias também são boas opções. A busca exaustiva apesar de garantir a solução ótima, de acordo com o critério utilizado, não é viável e pelos resultados obtidos, muitas vezes os subconjuntos de características encontrados não foram os melhores de acordo com o classificador C4.5 com *holdout*.

Nos grupos de exploração parcial, que utilizam algoritmos de Ramificação e Poda, o algoritmo BAB'^{CON} obteve solução equivalentes ao BAB original e, em muitos casos, com um menor tempo de processamento.

A partir dos estudos realizados durante a elaboração desta monografia, outros trabalhos podem ser realizados nas seguintes direções:

- Estudo de outros métodos para SC, complementando o estudo já realizado;
- Construção de Características;
- Discretização de Características;
- Extração de Características;
- Classificadores;
- Aplicações práticas do processo para Seleção de Características.

Referências Bibliográficas

[1] ALMUALLIM, H. S.; DIETTERICH, T. G. Learning with many irrelevant features. In: PROCEEDINGS OF THE 9TH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI'91), 1991. **Proceedings...** Anaheim, CA, USA: AAAI Press/MIT Press, 1991. p.547–552.

[2] ALMUALLIM, H. S.; DIETTERICH, T. G. Efficient algorithms for identifying relevant features. In: PROCEEDINGS OF THE 9TH CANADIAN CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1992. **Proceedings...** Vancouver, British Columbia: Morgan Kaufmann Publishers Inc., 1992. p.38–45.

[3] ALMUALLIM, H. S.; DIETTERICH, T. G. Learning boolean concepts in the presence of many irrelevant features. **Artificial Intelligence**, Essex, UK, v.69, n.1-2, p.279–305, Novembro, 1994.

[4] ASUNCION, A.; NEWMAN, D. **UCI Machine Learning Repository**. Disponível em http://www.ics.uci.edu/∼mlearn/MLRepository.html. Consultado em 16/11/2009.

[5] BLUM, A. L.; LANGLEY, P. Selection of relevant features and examples in machine learning. **Artificial Intelligence**, Essex, UK, v.97, n.1-2, p.245–271, Dezembro, 1997.

[6] BRODLEY, C. E.; UTGOFF, P. E. Multivariate decision trees. **Machine Learning**, Hingham, MA, USA, v.19, n.1, p.45–77, Abril, 1995.

[7] CHEN, X.-W. An improved branch and bound algorithm for feature selection. **Pattern Recognition Letters**, New York, NY, USA, v.24, n.12, p.1925–1933, Agosto, 2003.

[8] DASH, M.; LIU, H. Feature selection for classification. **Intelligent Data Analysis: An International Journal**, [S.l.], v.1, n.3, p.131–156, Março, 1997.

[9] DE CAMPOS, T. E. **Técnicas de Seleção de Características com Aplicações em Reconhecimento de Faces**. São Paulo, SP: Instituto de Matemática e Estatística da Universidade de São Paulo - IME-USP, Maio, 2001. Dissertação.

[10] DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification**. 2. ed. A Wiley-interscience Publication, 2000.

[11] FERRI, F. J.; KADIRKAMANATHAN, V.; KITTLER, J. Feature subset search using genetic algorithms. In: IEE/IEEE WORKSHOP ON NATURAL ALGORITHMS IN SIGNAL PROCESSING, 1993. **Proceedings...** Essex, UK: [s.n.], 1993.

[12] FERRI, F. J. et al. Comparative study of techniques for large-scale feature selection. In: Gelsema, E. S.; Kanal, L. N., editors, PATTERN RECOGNITION IN PRACTICE IV, 1994. **Proceedings...** Amsterdam, Holanda: Elsevier Science Inc., 1994. p.403–413.

[13] FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annual of Eugenics**, [S.l.], v.7, p.179–188, 1936.

[14] FRANÇOIS, J.-M. **jaDTi - Decision Trees: a Java implementation**. Disponível em http://www.run.montefiore.ulg.ac.be/∼francois/software/jaDTi/. Consultado em 16/11/2009.

[15] GUYON, I. Design of experiments for the nips 2003 variable selection benchmark. Julho, 2003. Relatório técnico .

[16] GUYON, I. et al. Result analysis of the nips 2003 feature selection challenge. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS), 2004. MIT Press, 2004. p.545–552.

[17] HAMAMOTO, Y. et al. Evaluation of the branch and bound algorithm for feature selection. **Pattern Recognition Letters**, New York, NY, USA, v.11, n.7, p.453–456, Julho, 1990.

[18] HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2. ed. Porto Alegre, RS: BOOKMAN, 2001.

[19] HUNT, E. B.; MARIN, J.; STONE, P. J. **Experiments in Induction**. Academic Press, 1966.

[20] JAIN, A. K.; ZONGKER, D. Feature selection: Evaluation, application, and small sample performance. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.19, n.2, p.153–158, Fevereiro, 1997.

[21] JAIN, A. K.; DUIN, R. P. W.; MAO, J. Statistical pattern recognition: A review. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Washington, DC, USA, v.22, n.1, p.4–37, Janeiro, 2000.

[22] JENSEN, R.; SHEN, Q. **Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches**. IEEE Press Series on Computational Intelligence. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2008.

[23] KIRA, K.; RENDELL, L. A. A practical approach to feature selection. In: Sleeman, D.; Edwards, P., editors, PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML'92), 1992. **Proceedings...** Aberdeen, Escócia, Reino Unido: Morgan Kaufmann Publishers Inc., 1992. p.249–256.

[24] KOHAVI, R. **Wrappers for Performance Enhancement and Oblivious Decision Graphs**. Stanford, CA, USA: Stanford University, 1995. Tese.

[25] KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. **Artificial Intelligence**, Essex, UK, v.97, n.1-2, p.273–324, Dezembro, 1997.

[26] KONONENKO, I. Estimating attributes: Analysis and extension of relief. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON MACHINE LEAR-

NING (ECML'94), 1994. **Proceedings...** Catania, Italy: Springer Berlin / Heidelberg, 1994. v.784 of **Lecture Notes in Computer Science**, p.171–182.

[27] KUDO, M.; SKLANSKY, J. Comparison of algorithms that select features for pattern classifiers. **Pattern Recognition**, New York, NY, USA, v.33, n.1, p.25–41, Janeiro, 2000.

[28] LEE, H. D. **Seleção e Construção de *Features* Relevantes para o Aprendizado de Máquina**. São Carlos, SP: Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo - ICMC-USP, Março, 2000. Dissertação.

[29] LEE, H. D. **Seleção de Atributos Importantes para a Extração de Conhecimento de Bases de Dados**. São Carlos, SP: Instituto de Ciências Matemáticas e de Coputação da Universidade de São Paulo - ICMC-USP, Dezembro, 2005. Tese.

[30] LIU, H.; SETIONO, R. A probabilistic approach to feature selection - a filter solution. In: PROCEEDINGS OF THE 13TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML'96), 1996. **Proceedings...** Bari, Itália: Morgan Kauffmann Publishers, 1996. p.319–327.

[31] LIU, H.; MOTODA, H. **Feature Selection for Knowledge Discovery and Data Mining**. Kluwer international series in engineering and computer science. 2. ed. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[32] LIU, H.; SETIONO, R. Scalable feature selection for large sized databases. In: PROCEEDINGS OF THE 4TH WORLD CONGRESS ON EXPERT SYSTEMS (WCES'98), 1998. **Proceedings...** Cidade do México, México: Morgan Kaufmann Publishers, 1998. p.521–528.

[33] MARTINS, C. A. **Uma abordagem para pré-processamento de dados textuais em algoritmos de aprendizado**. São Carlos, SP: Instituto de Ciências Matemáticas e de Coputação da Universidade de São Paulo - ICMC-USP, Agosto, 2003. Tese.

[34] MARTINS JR., D. C. **Redução de Dimensionalidade Utilizando Entropia Condicional Média Aplicada a Problemas de Bioinformática e de Processamento de Imagens**. São Paulo, SP: Instituto de Matemática e Estatística da Universidade de São Paulo - IME-USP, Dezembro, 2004. Dissertação.

[35] MONARD, M. C.; BARANAUSKAS, J. A. **Sistemas Inteligentes Fundamentos e Aplicações**, v.1, capítuloConceitos sobre Aprendizado de Máquina, p.39–56. Manole Ltda., Barueri, SP, 1. ed., 2003.

[36] NAKARIYAKUL, S.; CASASENT, D. P. Adaptive branch and bound algorithm for selecting optimal features. **Pattern Recognition Letters**, [S.l.], v.28, n.12, p.1415–1427, Setembro, 2007.

[37] NAKARIYAKUL, S.; CASASENT, D. P. Fast feature selection algorithm for poultry skin tumor detection in hyperspectral data. **Journal of Food Engineering**, [S.l.], v.94, n.3-4, p.358–365, Outubro, 2009.

[38] NARENDRA, P. M.; FUKUNAGA, K. A branch and bound algorithm for feature subset selection. **IEEE Transactions on Computers**, Washington, DC, USA, v.C-26, n.9, p.917–922, Setembro, 1977.

[39] PUDIL, P. et al. Floating search methods for feature selection with nonmonotoniccriterion functions. In: PROCEEDINGS OF THE 12TH CONFERENCE ON PATTERN RECOGNITION (ICPR'94), 1994. **Proceedings...** Jerusalem, Israel: [s.n.], 1994. v.2, p.279–283.

[40] PUDIL, P.; NOVOVIČOVÁ, J.; KITTLER, J. Floating search methods in feature selection. **Pattern Recognition Letters**, New York, NY, USA, v.15, n.11, p.1119–1125, Novembro, 1994.

[41] QUINLAN, J. R. Induction of decision tree. **Machine Learning**, Boston, USA, v.1, n.1, p.81–106, Março, 1986.

[42] QUINLAN, J. R. **C4.5: Programs for Machine Learning**. Morgan Kaufmann Publishers, 1993.

[43] QUINLAN, J. R. Improved use of continuous attributes in c4.5. **Journal of Artificial Intelligence Research (JAIR)**, [S.l.], v.4, p.77–90, Março, 1996.

[44] RONCATTI, M. A. **Avaliação de Métodos Ótimos e Subótimos de Seleção de Características de Texturas em Imagens**. São Carlos, SP: Instituto de Ciências Matemáticas e de Coputação da Universidade de São Paulo - ICMC-USP, Junho, 2008. Dissertação.

[45] SANTORO, D. M. **Sobre o Processo de Seleção de Subconjuntos de Atributos - As Abordagens Filtro e *Wrapper***. São Carlos, SP: Universidade Federal de São Carlos - UFSCar, Abril, 2005. Dissertação.

[46] SANTOS, D. P. D. **Seleção de Características: Abordagem Via Redes Neurais Aplicada à Segmentação de Imagens**. São Carlos, SP: Instituto de Ciências Matemáticas e de Coputação da Universidade de São Paulo - ICMC-USP, Março, 2007. Dissertação.

[47] SHANNON, C. E. A mathematical theory of communication. **Bell System Technical Journal**, [S.l.], v.27, p.379–423, 1948.

[48] SHANNON, C. E.; WEAVER, W. **The Mathematical Theory of Communication**. Urbana, IL, USA: The University of Illinois Press, 1949.

[49] SIEDLECKI, W. W.; SKLANSKY, J. On automatic feature selection. **International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)**, [S.l.], v.2, n.2, p.197–220, Junho, 1988.

[50] SLEPIAN, D. **Key Papers in the Development of Information Theory**. New York, NY, USA: IEEE Press, 1973.

[51] SLOANE, N. J. A.; WYNER, A. D. **Claude Shannon: Collected Papers**. New York, NY, USA: IEEE Press, 1993.

[52] SOMOL, P. et al. Adaptative floating search methods in feature selection. **Pattern Recognition Letters**, [S.l.], v.20, n.11-13, p.1157–1163, Novembro, 1999.

[53] SOMOL, P.; PUDIL, P. Oscillating search algorithms for feature selection. In: PROCEEDINGS OF 15TH INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION (ICPR'00), 2000. **Proceedings...** Barcelona, Espanha: [s.n.], 2000. v.2, p.406–409.

[54] SOMOL, P. et al. Fast branch & bound algorithm in feature selection. In: Sanchez, B.; Pineda, M. J.; J., W., editors, PROCEEDINGS OF THE 6TH WORLD MULTICONFERENCE ON SYSTEMIC, CYBERNETICS AND INFORMATICS (SCI'2000), 2000. **Proceedings...** Orlando, Florida, USA: [s.n.], 2000. p.646–651.

[55] SOMOL, P.; PUDIL, P.; GRIM, J. Branch & bound algorithm with partial prediction for use with recursive and non-recursive criterion forms. In: Singh, S.; Murshed, N.; Kropatsch, W., editors, PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON ADVANCES IN PATTERN RECOGNITION (ICAPR'2001), 2001. **Proceedings...** Rio de Janeiro, RJ: Springer Berlin / Heidelberg, 2001. Lecture Notes in Computer Science (LNCS), p.230–239.

[56] STEARNS, S. D. On selecting features for pattern classifiers. In: 3RD INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION (ICPR'76), 1976. **Proceedings...** Coronado, CA, USA: [s.n.], 1976. p.71–75.

[57] THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. 1. ed. San Diego, CA, USA: Academic Press, 1998.

[58] TOSCANI, L. V.; VELOSO, P. A. S. **Complexidade de algoritmos: análise, projeto e métodos**, v.1 of **Livros Didáticos**. 1. ed. Porto Alegre, RS: Editora Sagra Luzzatto, 2002.

[59] VOLTOLINI, R. **Projeto de um Ambiente para Classificação de Dados com Árvore de Decisão e Rede Neural Artificial Multicamada**. Cascavel, PR: Universidade Estadual do Oeste do Paraná - UNIOESTE, Novembro, 2008. Monografia.

[60] WU, X. et al. Top 10 algorithms in data mining. **Knowledge and Information Systems**, [S.l.], v.14, n.1, p.1–37, Janeiro, 2008.

[61] YEH, I.-C.; YANG, K.-J.; TING, T.-M. Knowledge discovery on rfm model using bernoulli sequence. **Expert Systems with Applications**, Tarrytown, NY, USA, v.36, n.3, p.5866–5871, Abril, 2009.

[62] YU, B.; YUAN, B. A more efficient branch and bound algorithm for feature selection. **Pattern Recognition**, New York, NY, USA, v.26, n.6, p.883–889, Junho, 1993.

[63] ZONGKER, D.; JAIN, A. K. Algorithms for feature selection: An evaluation. In: PROCEEDINGS OF THE 13TH INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION (ICPR'96) - VOLUME 2, 1996. **Proceedings...** Los Alamitos, CA, USA: IEEE Computer Society, 1996. v.2, p.18–22.