Computação evolutiva para a construção de regras de conhecimento com propriedades especificas

Adriano Donizete Pila

Computação evolutiva para a construção de regras de conhecimento com propriedades especificas

Adriano Donizete Pila

Orientadora: ***Profa. Dra. Maria Carolina Monard***

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional.

"VERSÃO REVISADA APÓS A DEFESA"

Data da Defesa:	12/04/2007
Visto do Orientador:	

USP – São Carlos
Maio/2007

Este documento foi preparado com o formatador de textos LaTeX. Foi utilizado um estilo (*style*) desenvolvido por Ronaldo Cristiano Prati. O sistema de citações de referências bibliográficas utiliza o padrão *Apalike* do sistema BibTeX. Algumas abreviações utilizadas neste trabalho não foram traduzidas da língua inglesa para a portuguesa por serem amplamente conhecidas e difundidas na comunidade acadêmica. Todos os endereços de *Internet* utilizados nas referências bibliográficas tiveram seu último acesso no mês de janeiro/2007.

"É verdade que não podemos encontrar a pedra filosofal, mas é bom que ela seja procurada. Procurando-a, encontramos muitos segredos que não procurávamos".

Bernard le Bovier de Fontenelle (1657–1757)

À minha esposa, Cassia, pelo apoio, paixão e dedicação, sem os quais este trabalho não teria sentido.

Aos meus filhos, Heitor e Enzo, pelo intenso afeto e descontração nos momentos mais difíceis.

Aos meus pais que sempre souberam valorizar o estudo e a contínua formação.

Agradecimentos

Entendo que nos agradecimentos devemos ser o mais espontâneos possível, reconhecendo e deixando registradas as colaborações que de alguma forma foram imprescindíveis para a realização do trabalho. No caso deste trabalho, uma **TESE** de doutorado, foram longos 5 anos se dedicando aos estudos, ao trabalho e à família. Foi muito difícil, muito mesmo, mas se estou escrevendo isso é porque deu certo. Existem inúmeras pessoas cujos nomes devo deixar gravados no papel pela sua valorosa contribuição.

Agradeço primeiramente à Deus que com Sua providência me deu forças, perseverança e sanidade para não desistir de meu objetivo. Sem Ele eu não teria conseguido, porque tudo pode quem nele acredita.

Agradeço aos amigos do LABIC pela ajuda incondicional de muitos. Devo destacar a presteza do Ronaldo Prati que sempre têm algo à colaborar ou ajudar no desenvolvimento do trabalho. Agradeço à Flávia Bernardini pela força e troca de idéias. Agradeço ao Richardson Voltolini, que disponibilizou parte do que ele desenvolveu no mestrado para gerar alguns gráficos. Agradeço especialmente ao Rafael Giusti que trabalhou neste projeto como aluno de iniciação científica e muito colaborou no projeto e desenvolvimento da biblioteca. Definitivamente, sem o seu comprometimento e inteligência, funcionalidades da biblioteca não existiriam.

Agradeço imensamente aos meus pais, Antonio e Ivete Pila, os quais sempre souberam dar importância à formação técnica, intelectual e do caráter. Ninguém é alguém, se não tiver caráter. Seus ensinamentos e exemplos de perseverança e honestidade sempre me acompanharão.

Quero agradecer aos funcionários do Instituto de Ciências Matemática e de Computação da Universidade de São Paulo pelo trabalho dedicado que todos desenvolvem, o que facilita, e muito, o trabalho dos alunos da graduação e pós-graduação. Agradeço também aos funcionários da biblioteca que sempre

foram compreensíveis com o fato de eu morar em outra cidade.

Finalmente, mas não menos importante, quero agradecer às pessoas que serviram de motivação para a realização e conclusão deste trabalho. Embora eu seja muito perseverante, sem essas pessoas eu certamente teria desistido de meu objetivo perante os vários percalços ocorridos nestes últimos anos.

Agradeço à minha **ORIENTADORA**, Profa. Dra. Maria Carolina Monard por esses longos oito anos de convivência. A conheci em 1999 quando do início do mestrado. Aqui devo agradecer ao Prof. Dr. Ricardo Luís de Freitas que me apresentou-a. Nos dois anos de mestrado foi um aprendizado significativo, cresci profissionalmente e aprendi que caráter também é fundamental na pesquisa. Nos anos seguintes, já no doutorado, aprendi que sucesso se conquista com trabalho sério, comprometido, árduo, direcionado e bem conduzido. A Profa. Carolina é um exemplo de profissional a ser seguido, admirada e à qual devo muito trabalho e respeito. Meus sinceros agradecimentos por permitir que eu fosse seu orientado. As pessoas precisam enxergar o valor que uma ótima orientadora e pessoa possuem e como isso faz diferença no trabalho e na vida.

Principalmente, agradeço à minha **FAMÍLIA**, motivo de meu trabalho e minhas paixões: minha esposa, Cássia Pila, e meus filhos Heitor e Enzo Pila. Minha família é minha maior motivação. O apoio e incentivo dados pela minha esposa foram fundamentais para a conclusão deste trabalho. O seu carinho e desejo de superação muito me ajudaram a superar as dificuldades. Olhar meus filhos e ver como somos impotentes perante aquela forma carinhosa de me chamar de papai, reaviva a chama da luta e nos faz lembrar que nada na vida é de graça. Tudo requer esforço.

Complementando, nessa versão revisada após a defesa, gostaria de agradecer aos membros da banca examinadora que muito contribuíram para o polimento dessa versão. Assim, deixo aqui registrado meus agradecimentos aos professores: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho, Profa. Dra. Aurora Trinidad Ramirez Pozo, Profa. Dra. Heloisa de Arruda Camargo, Profa. Dra. Luisa Fernanda Ribeiro Reis e Profa. Dra. Maria Carolina Monard.

> Temos de fazer o melhor que podemos. Esta é a nossa sagrada responsabilidade humana.
> Albert Einstein (1561–1626)

Resumo

A maioria dos algoritmos de aprendizado de máquina simbólico utilizam regras de conhecimento *if-then* como linguagem de descrição para expressar o conhecimento aprendido. O objetivo desses algoritmos é encontrar um conjunto de regras de classificação que possam ser utilizadas na predição da classe de novos casos que não foram vistos a priori pelo algoritmo. Contudo, este tipo de algoritmo considera o problema da interação entre as regras, o qual consiste na avaliação da qualidade do conjunto de regras induzidas (classificador) como um todo, ao invés de avaliar a qualidade de cada regra de forma independente. Assim, como os classificadores têm por objetivo uma boa precisão nos casos não vistos, eles tendem a negligenciar outras propriedades desejáveis das regras de conhecimento, como a habilidade de causar surpresa ou trazer conhecimento novo ao especialista do domínio. Neste trabalho, estamos interessados em construir regras de conhecimento com propriedades específicas de forma isolada, *i.e.* sem considerar o problema da interação entre as regras. Para esse fim, propomos uma abordagem evolutiva na qual cada individuo da população do algoritmo representa uma única regra e as propriedades específicas são codificadas como medidas de qualidade da regra, as quais podem ser escolhidas pelo especialista do domínio para construir regras com as propriedades desejadas. O algoritmo evolutivo proposto utiliza uma rica estrutura para representar os indivíduos (regras), a qual possibilita considerar uma grande variedade de operadores evolutivos. O algoritmo utiliza uma função de aptidão multi-objetivo baseada em *ranking* que considera de forma concomitante mais que uma medida de avaliação de regra, transformando-as em uma função simples-objetivo. Como a avaliação experimental é fundamental neste tipo de trabalho, para avaliar nossa proposta foi implementada a *Evolutionary Computing Learning Environment* — $\mathcal{ECLE}$ — que é uma biblioteca de classes para executar e avaliar o algoritmo evolutivo sob diferentes cenários. Além

disso, a $\mathcal{ECLE}$ foi desenvolvida considerando futuras implementações de novos operadores evolutivos. A $\mathcal{ECLE}$ está integrada ao projeto DISCOVER, que é um projeto de pesquisa em desenvolvimento em nosso laboratório para a aquisição automática de conhecimento. Analises experimentais do algoritmo evolutivo para construir regras de conhecimento com propriedades específicas, o qual pode ser considerado uma forma de análise inteligente de dados, foram realizadas utilizando a $\mathcal{ECLE}$. Os resultados mostram a adequabilidade da nossa proposta.

Abstract

Most symbolic machine learning approaches use if-then knowledge rules as the description language in which the learned knowledge is expressed. The aim of these learners is to find a set of classification rules that can be used to predict new instances that have not been seen by the learner before. However, these sorts of learners take into account the rule interaction problem, which consists of evaluating the quality of the set of rules (classifier) as a whole, rather than evaluating the quality of each rule in an independent manner. Thus, as classifiers aim at good precision to classify unseen instances, they tend to neglect other desirable properties of knowledge rules, such as the ability to cause surprise or bring new knowledge to the domain specialist. In this work, we are interested in building knowledge rules with specific properties in an isolated manner, i.e. not considering the rule interaction problem. To this end, we propose an evolutionary approach where each individual of the algorithm population represents a single rule and the specific properties are encoded as rule quality measure, a set of which can be freely selected by the domain specialist. The proposed evolutionary algorithm uses a rich structure for individual representation which enables one to consider a great variety of evolutionary operators. The algorithm uses a ranking-based multi-objective fitness function that considers more than one rule evaluation measure concomitantly into a single objective. As experimentation plays an important role in this sort of work, in order to evaluate our proposal we have implemented the Evolutionary Computing Learning Environment — $\mathcal{ECLE}$ — which is a framework to evaluate the evolutionary algorithm in different scenarios. Furthermore, the $\mathcal{ECLE}$ has been implemented taking into account future development of new evolutionary operators. The $\mathcal{ECLE}$ is integrated into the DISCOVER project, a major research project under constant development in our laboratory for automatic knowledge acquisition and analysis. Experimental analysis of the

evolutionary algorithm to construct knowledge rules with specific properties, which can also be considered an important form of intelligent data analysis, was carried out using $\mathcal{ECLE}$. Results show the suitability of our proposal.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Regras

Capítulo

1

INTRODUÇÃO

Cada nova descoberta na ciência têm a função de uma pequena peça de retalho, a qual pertence a uma bela, colorida, grande e inacabada colcha. Nós, seres humanos, tentamos em vão terminá-la, como uma forma de entendermos nossa própria existência.
Pierre-Simon Laplace (1749–1827)

Com o irrefreável crescimento da quantidade de informações armazenadas em bases de dados por organizações de todo o mundo, tem-se tornado cada vez mais importante o uso de técnicas computacionais para a compreensão dessas informações. Na área de Inteligência Artificial (IA), os algoritmos de Aprendizado de Máquina (AM) permitem automatizar o processo de extração do conhecimento implícito nessas bases de dados (Han & Kamber, 2006).

A maioria dos algoritmos de aprendizado recebem como entrada um conjunto de dados no formato atributo-valor. Nesse formato, cada exemplo observado (também chamado de caso ou experiência) é descrito em uma linha dessa tabela, a qual contém as características que descrevem aquela experiência. No caso dos exemplos estarem rotulados, podem ser utilizados algoritmos de aprendizado supervisionado, e mais especificamente, algoritmos simbólicos.

A maioria dos algoritmos de aprendizado de máquina simbólico utilizam regras de conhecimento *if-then* como linguagem de descrição para expressar o conhecimento aprendido. O objetivo desses algoritmos é encontrar um conjunto de regras de classificação que possam ser utilizadas na predição da classe de novos casos que não foram vistos a priori pelo algoritmo. Contudo, este tipo de algoritmo considera o problema da interação entre as regras, o qual consiste na avaliação da qualidade do conjunto de regras induzidas (classificador) como um todo, ao invés de avaliar a qualidade de cada regra de forma independente. Assim, como os classificadores têm por objetivo uma boa precisão nos casos não vistos, eles tendem a negligenciar outras propriedades desejáveis das regras de conhecimento, como a habilidade de causar surpresa ou trazer conhecimento novo ao especialista do domínio.

Quando o objetivo é encontrar regras de conhecimento com propriedades específicas, uma possível abordagem é induzir um classificador utilizando algum algoritmo de aprendizado simbólico que induza um conjunto de regras de conhecimento e desse conjunto de regras, é feita uma seleção em busca de regras que apresentem as propriedades desejadas (Freitas, 1998b). Entretanto, essa abordagem nem sempre surtirá os resultados esperados, uma vez que o objetivo desses algoritmos é construir um conjunto de regras de conhecimento, as quais interagem entre si, que cubram bem o conjunto de exemplos de treinamento utilizado para construir essas regras, bem como tenham um bom poder de predição da classe (rótulo) de novos exemplos. Esse objetivo dos algoritmos de aprendizado é atingido utilizando medidas específicas para esse fim, sem considerar medidas que exploram outras propriedades das regras. Neste trabalho, estamos interessados em construir regras de conhecimento com propriedades específicas de forma isolada, ou seja, sem considerar o problema de interação entre as regras. Uma forma de construir essas regras é utilizando algoritmos evolutivos, os quais têm sido pouco explorados para esse fim.

Algoritmos evolutivos podem ser utilizados nas mais diversas áreas em que haja demanda por um processo de otimização, o qual é guiado por uma função objetivo que se deseja otimizar. O uso de algoritmos evolutivos desperta interesse por conta da forma global de explorar o espaço de soluções (Bäck et al., 2000). Na construção de regras de conhecimento com propriedades específicas o que se deseja é criar regras que apresentem as propriedades especificadas pelo especialista do domínio. A presença dessas propriedades nas regras pode ser avaliada pela função de aptidão que considera diferentes medidas de avaliação de regras de conhecimento. Nessa abordagem, os indivíduos do algoritmo evolutivo representam regras de conhecimento, aos quais são aplicados operadores evolutivos que guiam o processo adaptativo na busca de

regras que melhor satisfaçam a função de aptidão.

Na descoberta de regras de conhecimento individuais existe a necessidade de construir regras que possuam algumas propriedades desejáveis, como novidade, por exemplo. Entretanto, existe também a necessidade que a regra construída cubra vários exemplos, ou seja, o conhecimento novo expresso por aquela regra se verifica em um subconjunto expressivo dos exemplos do domínio. Assim, a regra de conhecimento construída deve possuir boa cobertura e, ainda, apresentar as propriedades especificadas pelo especialista. Porém, essas propriedades não podem ser consideradas independentemente, pois há uma interação entre elas. Dessa forma, o que se espera é atingir um equilíbrio entre elas.

Na área de otimização multi-objetivo são pesquisados métodos para encontrar soluções ótimas, as quais representam uma relação de melhor compromisso entre os diversos objetivos considerados (Coello et al., 2002). Nesse contexto, os algoritmos evolutivos multi-objetivos tem sido freqüentemente utilizados em diversas áreas (Deb, 2001). No entanto, o uso de algoritmos evolutivos multi-objetivo para a descoberta de regras de conhecimento ainda é pouco explorada, com poucas referências encontradas na literatura (Setzkorn & Paton, 2003; Ghosh & Nath, 2004; Smaldon & Freitas, 2006). Na aplicação de algoritmos evolutivos para construir regras com medidas multi-objetivo pode ser feita uma simplificação do problema multi-objetivo de forma a transformá-lo em um problema simples-objetivo (Freitas, 1998b; Ishibuchi et al., 2001; Ishibuchi & Yamamoto, 2004; Romao et al., 2004).

A construção de regras de conhecimento constitui um problema interessante, já que existe uma vasta gama de tipos de operadores evolutivos que podem ser propostos, pois os mesmos dependem muito da representação adotada. Dessa forma, o campo da construção de regras de conhecimento com propriedades específicas utilizando algoritmos evolutivos é bastante amplo, motivando assim o desenvolvimento deste trabalho. A seguir são apresentados os objetivos deste trabalho e as suas principais contribuições.

1.1 Abordagem Proposta e Principais Contribuições

A proposta deste trabalho consiste em investigar o uso de algoritmos evolutivos como meio de construir regras de conhecimento, as quais devem apresentar propriedades desejáveis especificadas pelo usuário. Dentre essas propriedades desejáveis além daquelas usuais, como cobertura e precisão, espera-se

que as regras construídas retratem conhecimento novo, raro e que cause surpresa ao especialista do domínio.

Neste trabalho, propomos um algoritmo evolutivo multi-objetivo para a construção de regras com propriedades especificadas pelo usuário. Esse algoritmo pressupõe como entrada um conjunto de regras quaisquer que tenham relação com o domínio do conjunto de dados. Essas regras podem ser aquelas presentes em classificadores simbólicos induzidos por algoritmos de aprendizado, regras de associação ou mesmo regras fornecidas pelo especialista do domínio. Essas regras são utilizadas para povoar a população inicial do algoritmo evolutivo por nós proposto.

A manipulação de regras de conhecimento simbólicas enquanto indivíduos de um algoritmo evolutivo demanda por operadores evolutivos específicos, os quais dependem da representação dos indivíduos. O algoritmo evolutivo proposto utiliza uma rica estrutura para representar os indivíduos (regras), a qual possibilita considerar uma grande variedade de operadores evolutivos (*crossover* e mutação).

A avaliação de regras de conhecimento empregada neste trabalho está baseada no trabalho de Lavrač et al. (1999), no qual é proposto uma forma de padronização de diversas medidas de avaliação de regras em um mesmo *framework*. A função de aptidão pode ser composta por uma única medida de avaliação de regras (avaliação simples-objetivo) ou por um conjunto de medidas (avaliação multi-objetivo). No caso da avaliação ser multi-objetivo, propomos várias formas de combinar esses objetivos em uma função multi-objetivo. Uma delas é por meio do uso de *rankings*.

A avaliação experimental é essencial no contexto deste trabalho. Dessa forma, foi desenvolvida uma biblioteca de classes chamada Evolutionary Computing Learning Environment — $\mathcal{ECLE}$ — que implementa o algoritmo evolutivo com métodos de seleção, operadores evolutivos e funções de avaliação especificamente projetados para a construção de regras de conhecimento com propriedades específicas. Também faz parte da proposta deste trabalho um ambiente de execução de experimentos e coleta de resultados chamado SNIFFERECLE, capaz de automatizar a execução de um grande número de experimentos, gerando informações detalhadas, tabelas e gráficos dos resultados obtidos. A biblioteca de classes desenvolvida faz parte de um projeto maior em desenvolvimento no nosso laboratório — LABIC — para implementar métodos de pré-processamento e aquisição automática de conhecimento, o projeto DISCOVER. Vale ressaltar que este trabalho relacionado com uma abordagem evolutiva para a construção de regras de conhecimento com propriedades especificas é o primeiro desenvolvido em nosso grupo de pesquisa. Analises experimen-

tais do algoritmo evolutivo proposto foram realizadas utilizando a $\mathcal{ECLE}$. Os resultados mostram a adequabilidade da nossa proposta.

Consideramos que a estrutura para representar os indivíduos (regras), a qual permite utilizar uma ampla variedade de operadores evolutivos, a avaliação de indivíduos, o uso de *rankings* para tratar funções multi-objetivo, bem como a implementação da $\mathcal{ECLE}$ e o SNIFFERECLE, constituem uma contribuição à área de descoberta de conhecimento e análise inteligente de dados. Essas contribuições originaram as publicações descritas brevemente a seguir.

Em Pila et al. (2006a) é descrita a arquitetura geral da $\mathcal{ECLE}$, mostrada a forma de representação de regras de conhecimento no contexto do algoritmo evolutivo proposto e são descritas as implementações dos principais operadores evolutivos de *crossover* e mutação, bem como algumas funções de avaliação simples-objetivo.

Em Pila et al. (2006b) e Pila et al. (2006c) é descrito o algoritmo evolutivo utilizando uma função de avaliação multi-objetivo, a qual pode ser uma combinação de diversas medidas de avaliação de regras de conhecimento. Nesses trabalhos, essa combinação é feita utilizando *ranking* como forma de promover um melhor equilíbrio entre as medidas que participam da função de avaliação multi-objetivo.

Em Giusti et al. (2006) é descrito todo o projeto e implementação, bem como todas as facilidades providas pela $\mathcal{ECLE}$. São também fornecidos exemplos da biblioteca de classes implementada. Esse documento é importante por servir de base para futuras expansões do trabalho já desenvolvido.

1.2 Organização do Trabalho

Esta Tese está organizada em sete capítulo, descritos a seguir:

Capítulo 2 — Aprendizado de Máquina. Nesse capítulo são apresentados os conceitos de aprendizado de máquina importantes para o desenvolvimento deste trabalho, mais especificamente o aprendizado simbólico supervisionado. É apresentado o formato da tabela que descreve os exemplos, bem como a linguagem de descrição do conhecimento extraído dos dados. Ainda, é apresentada a matriz de contingência que serve de base para a definição das medidas individuais de avaliação de regras.

Capítulo 3 — Computação Evolutiva. Os conceitos sobre computação evolutiva e as áreas envolvidas são brevemente descritos nesse capítulo. É

apresentada uma visão geral dos processos que compõem um algoritmo evolutivo, descrevendo o funcionamento dos métodos de seleção, operadores evolutivos e da função de avaliação.

Capítulo 4 — Algoritmo Evolutivo Proposto. O algoritmo evolutivo por nós proposto para a construção de regras de conhecimento com propriedades específicas é descrito nesse capítulo. É apresentado o formato padrão de regras que é utilizado para representar as regras de conhecimento fornecidas como entrada do algoritmo evolutivo. A representação das regras de conhecimento, por nós proposta, como indivíduos do algoritmo evolutivo e os operadores evolutivos que atuam sobre essa estrutura são todos detalhadamente explanados. O uso de *rankings* para compor uma função multi-objetivo que considera diferentes medidas de avaliação de regras também é tema desse capítulo. Finalmente, é apresentado o critério utilizado para testar a convergência do algoritmo evolutivo.

Capítulo 5 — Projeto da Biblioteca Evolutionary Computing Learning Environment. A arquitetura geral do sistema que implementa o algoritmo evolutivo, bem como a hierarquia de classes é tema desse capítulo. É mostrado como essa implementação se relaciona com outras implementações realizadas para o projeto DISCOVER. Os possíveis parâmetros que o algoritmo evolutivo proposto pode assumir e como esses parâmetros podem ser configurados utilizando uma linguagem por nós proposta, bem como um exemplo de uso do algoritmo evolutivo, são nesse capítulo mostrados. Também é descrito o ambiente SNIFFERECLE proposto como forma de automatizar a execução do algoritmo utilizando configurações diferentes e coleta dos resultados gerados.

Capítulo 6 — Avaliação Experimental. Nesse capítulo são mostrados os resultados experimentais obtidos sobre diversos conjuntos de dados e cenários diferentes. Os resultados mostram que o algoritmo evolutivo proposto é promisor, sendo possível construir regras de conhecimento com propriedades específicas e que apresentam um equilíbrio entre as medidas de avaliação de regras consideradas.

Capítulo 7 — Conclusão. Nesse capítulo é apresentada a conclusão deste trabalho, apresentando os resultados obtidos e as limitações da proposta, bem como possíveis trabalhos futuros.

Capítulo

2

APRENDIZADO DE MÁQUINA

O problema não é quanto as máquinas pensam, mas sim quando os homens pensam.
B. F. Skinner (1904–1990), *Contingência do Reforço*

2.1 Considerações Iniciais

DEVIDO aos avanços tecnológicos, atualmente é possível armazenar, acessar e processar grandes volumes de dados. Ainda que o processo que gera esses dados não seja completamente conhecido, é possível identificar um processo que explica os dados observados, já que eles não são gerados aleatoriamente. Pode ser que não seja possível identificar exatamente o processo que explica os dados observados, porém, pode-se tentar construir uma boa aproximação do mesmo tal que seja capaz de explicar parte desses dados por meio da identificação de certos padrões ou regularidades. Esses padrões podem ser úteis para entender o processo ou, sob certas condições, fazer predições. Esse é o um dos objetivos do Aprendizado de Máquina, uma sub-área de Inteligência Artificial, que procura otimizar

os parâmetros de um modelo que aproxima o processo que explica os dados (exemplos) utilizando exemplos ou experiências passadas observadas. Esse modelo pode ser *preditivo*, *i.e* com capacidade de realizar predições futuras, *descritivo*, para descrever ou ganhar conhecimento dos dados, ou ambos. Devido ao fato de AM realizar inferências utilizando um subconjunto (amostra) dos dados, ela apóia-se na estatística para construir esses modelos (Alpaydin, 2004). Algoritmos de aprendizado podem ser agrupados de maneiras diferentes segundo diversos critérios. Neste trabalho usamos o critério do grau de supervisão presente nos dados de treinamento (amostra) disponíveis para o aprendizado.

2.2 Descrição de Exemplos

Diversas linguagens podem ser utilizadas para representar os exemplos usados pelos algoritmos de aprendizado, as quais apresentam diferente complexidade e poder de descrição. Uma dessas linguagens, a qual apresenta poder de descrição equivalente ao da lógica proposicional, é amplamente utilizada pela maioria dos algoritmos de aprendizado. Nessa linguagem, utilizada neste trabalho, os exemplos são descritos utilizando um conjunto de atributos apropriados, os quais podem assumir valores diferentes, e cada exemplo é representado em uma linha de uma tabela chamada tabela atributo-valor. Essa representação é praticamente onipresente, seja nas transações bancárias, nos registros escolares, nas compras de supermercado, entre muitos outros. Ela tem sido usada para armazenar milhares de terabytes de dados gerados diariamente, os quais povoam inúmeros bancos de dados.

Na Tabela 2.1 é apresentado o formato geral de uma tabela atributo-valor com N exemplos E_i com $i = 1, ..., N$, na forma $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$. Os $\mathbf{x}_i$ são vetores da forma $(x_{i1}, x_{i2}, ..., x_{iM})$, com valores discretos ou contínuos. Assim, x_{ij} refere-se ao valor do atributo j, denominado $\mathbf{X}_j$, do exemplo E_i, como mostrado na Tabela 2.1[1]. Os valores y_i, os quais podem ou não existir, referem-se ao valor do atributo $\mathbf{Y}$, denominado atributo *classe*. No caso de classificação, os valores do atributo classe $\mathbf{Y}$ pertencem a um conjunto C de classes discretas C_v com $v = 1, ..., N_{Cl}$, ou seja, $C = \{C_1, ..., C_{N_{Cl}}\}$.

Um conjunto de exemplos para o qual é fornecido o atributo classe é denominado conjunto de exemplos *rotulados*, caso contrario de *não rotulados*. A classe é o conceito-meta que descreve o fenômeno de interesse. Por exemplo,

[1] Caso o valor de x_{ij} não seja especificado (valor desconhecido), o símbolo "?" é atribuído à x_{ij}.

	X_1	X_2	$\ldots$	X_M	Y
E_1	x_{11}	x_{12}	$\vdots$	x_{1M}	y_1
E_2	x_{21}	x_{22}	$\vdots$	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	$\vdots$	x_{NM}	y_N

Tabela 2.1: Exemplos no formato atributo valor

em uma base de dados médica, o conceito classe pode ser *doente* ou *não-doente*, enquanto que em uma base de mineração de textos o conceito classe pode ser o assunto ao qual o texto está relacionado, tal como *economia*, *esporte*, *política*, *informática* ou *ciência*.

Dado um conjunto de exemplos, esse conjunto de exemplos disponível deve ser utilizado em diferentes fases do aprendizado. Dependendo da fase, o subconjunto do conjunto de exemplos original utilizado é identificado como descrito a seguir:

- *conjunto de treinamento*: esse conjunto é a principal entrada dos algoritmos de aprendizado. É a partir dele que são construídos os modelos e, portanto, ele deve ser representativo da distribuição da população para realizar inferência sobre esses dados. Na literatura, esse conjunto também é conhecido como *seen cases*, pois refere-se aos exemplos que foram "vistos" pelo algoritmo de aprendizado durante a construção do modelo.

- *conjunto de teste*: esse conjunto é utilizado para avaliar o modelo construído. Esse conjunto, também conhecido como *unseen cases*, não deve ser apresentado ao algoritmo de aprendizado durante a construção do modelo. Idealmente, esse conjunto não deveria ter exemplos em comum com o conjunto de treinamento.

- *conjunto de validação*: em alguns casos, pode ser necessário utilizar exemplos para realizar ajustes no modelo construído pelo algoritmo de aprendizado. Esses exemplos não são utilizados diretamente na construção do modelo, mas são utilizados para o seu ajuste. Dessa maneira, esses exemplos são indiretamente "vistos" durante o processo de aprendizado, o que obriga que os exemplos de validação sejam distintos dos de teste.

Muitos algoritmos de aprendizado têm sido propostos na literatura, os quais podem ser agrupados utilizando diversos critérios. Um desses critérios é o

grau de supervisão presente nos dados, bem como a linguagem na qual é expresso o modelo. Este trabalho concentra-se em dados rotulados com um conjunto $\boldsymbol{C}$ de classes discretas — Seção 2.2 — utilizando regras de conhecimento como linguagem de descrição.

2.3 Aprendizado Supervisionado

No aprendizado supervisionado é fornecido ao algoritmo, denominado de *indutor*, um conjunto de exemplos de treinamento $E = \{E_1, E_2..., E_N\}$ no qual cada exemplo E_i tem o valor y_i do atributo classe Y associado, ou seja, $E_i = (\mathbf{x}_i, y_i)$ — Seção 2.2 — tal que $y_i = f(\mathbf{x}_i)$ para uma função f desconhecida, denominada de *função conceito*. O objetivo do aprendizado supervisionado é encontrar ou aproximar essa função conceito f. Entretanto, os exemplos de treinamento utilizados no processo são geralmente insuficientes para caracterizar f, assim, o objetivo dos algoritmos consiste em induzir uma boa aproximação h de f tal que $h(\mathbf{x}) \approx f(\mathbf{x})$. Nesse caso h é chamada de hipótese (ou modelo) sobre f. Em outras palavras, do conjunto de possíveis hipóteses $\boldsymbol{H}$ que aproximam f, o algoritmo deve encontrar uma boa hipótese $h \in \boldsymbol{H}$ a qual deve generalizar bem, ou seja, deve predizer corretamente o valor da classe de exemplos não vistos durante a sua construção. Esse é o problema fundamental de indução (Russell & Norvig, 2003).

Se o atributo classe Y é quantitativo, *i.e.*, pode assumir valores reais, a hipótese h é denominada *regressor* e a tarefa de aprendizado é chamada *regressão*. Se o atributo classe é nominal, *i.e.*, pode assumir um conjunto finito de valores $\{C_1, ..., C_{N_{Cl}}\}$, a hipótese h é denominada *classificador* e a tarefa de aprendizado é chamada *classificação*. Mais especificamente, neste trabalho estamos interessados em conhecimento descrito em uma linguagem simbólica, apresentada a seguir.

2.4 Terminologia e Linguagem de Descrição

Os conceitos induzidos por algoritmos de aprendizado simbólicos são geralmente representados por árvores de decisão ou conjuntos de regras. Como sempre é possível escrever uma árvore de decisão como um conjunto de regras disjuntas, deste ponto em diante o termo *regra* refere-se a uma regra extraída de uma árvore de decisão (regras disjuntas) ou uma regra diretamente induzida por um algoritmo de aprendizado simbólico supervisionado (Monard

& Baranauskas, 2003). Uma regra é geralmente representada na forma R: *if* <*complexo*> *then* <class> = C_i, na qual C_i é um dos possíveis valores da classe, *i.e.* $C_i \in \{C_1, C_2, ..., C_{Ncl}\}$ e <*complexo*> é uma disjunção de conjunções de condições para os atributos da forma X_i *op* valor, com $X_i \in \boldsymbol{X}$ e $op \in \{<, \leq, >, \geq, =\}$. Qualquer regra

$$R : \underbrace{\textbf{\textit{if}} \ < \textit{complexo} >}_{Body \text{ ou } B} \ \textbf{\textit{then}} \ \underbrace{\text{<class> = } C_i}_{Head \text{ ou } H}$$

pode ser genericamente representada por $Body \rightarrow Head$ ou, resumidamente, $B \rightarrow H$. Assim, diz-se que um exemplo E_i é coberto por uma regra R, se e somente se o exemplo satisfaz todas as condições da regra. Ou seja, um exemplo E_i é coberto por uma regra R, se e somente se B é verdade. Com base nas possíveis coberturas em relação às condições e ao conseqüente da regra, pode-se definir a matriz de contingência de uma regra — Seção 2.6.

2.5 Matriz de Confusão

Após um classificador ser induzido ele pode ser avaliado de várias formas. Uma delas, freqüentemente utilizada, é a taxa de erro ou, de forma complementar, a precisão do classificador. Essa taxa informa qual a estimativa do percentual de acertos na predição da classe de novos exemplos. Essa taxa é obtida a partir da matriz de confusão, que informa quantos erros e acertos ocorreram na predição da classe dos exemplos.

Considerando uma classificação binária, com dois valores para o atributo classe, $(+)$ e $(-)$, na predição da classe de cada exemplo o classificador pode predizer corretamente ou incorretamente a classe do exemplo. Existem 4 possibilidades:

1. *Falso negativo (Fn)*, quando o exemplo pertence à classe $(+)$ e é predito pelo classificador como pertencente à classe $(-)$.
2. *Falso positivo (Fp)*, quando o exemplo pertence à classe $(-)$ e é predito pelo classificador como pertencente à classe $(+)$.
3. *Verdadeiro negativo (Tn)*, quando o exemplo pertence à classe $(-)$ e é predito pelo classificador como pertencente à classe $(-)$.
4. *Verdadeiro positivo (Tp)*, quando o exemplo pertence à classe $(+)$ e é predito pelo classificador como pertencente à classe $(+)$.

Considerando um valor da classe como $(+)$ e os outros valores como $(-)$, pode-se generalizar essas definições da matriz de confusão para um classificador com quaisquer número de classes. Na Tabela 2.2 é mostrada a matriz de confusão para o exemplo da classificação binária, a qual pode ser generalizada para um classificador com qualquer número de classes (Baranauskas & Monard, 2000).

		Predita (+)	**Predita** (-)	**Precisão da classe**	**Precisão Total**
Verdadeira	(+)	*Tp*	*Fn*	$\frac{Tp}{Tp+Fn}$	$\frac{Tp+Tn}{N}$
	(-)	*Fp*	*Tn*	$\frac{Tn}{Fp+Tn}$	

Tabela 2.2: Matriz de confusão – classificação binária

A precisão de um classificador é uma medida importante na predição da classe de novos exemplos. No entanto, existem casos para os quais a precisão do classificador não é boa, mas o conhecimento induzido, *i.e.* conjunto de regras, pode conter regras que individualmente possuam uma boa precisão, ou mesmo possuam algum outro tipo de propriedade interessante da regra. Cada regra pode ser avaliada individualmente com o uso da matriz de contingência, apresentada a seguir.

2.6 Matriz de Contingência

Com base nas possíveis coberturas em relação às condições e ao conseqüente da regra, pode-se definir a matriz de contingência de uma regra, a qual é uma generalização da matriz de confusão do classificador. Em outras palavras, a matriz de confusão refere-se ao classificador como um todo, ou seja, o classificador é tratado como uma caixa preta, enquanto que a matriz de contingência é calculada para qualquer regra individual.

Dados uma regra R, e um exemplo $E_i = (\mathbf{x}_i, y_i)$ com a sua respectiva classe y_i, pode-se aplicar a regra ao exemplo e comparar o resultado previsto pela cabeça H da regra com a verdadeira classe y_i do exemplo. Aplicando a regra a um conjunto de teste — Tabela 2.1 na página 9 — que contenha N exemplos, podemos derivar, para cada regra, a sua matriz de contingência — Tabela 2.3. A matriz de contingência também pode ser representada usando freqüências relativas, como mostrado na Tabela 2.4, na qual os valores foram divididos por N, onde N é o número de exemplos considerado para avaliar a regra, ou seja $f_\epsilon = \frac{\epsilon}{N}$. Neste trabalho, consideramos a freqüência relativa $\frac{\epsilon}{N}$, associada

ao evento ϵ, como uma estimativa de probabilidade para o evento ϵ, denotada $P(\epsilon)$.

	H	$\overline{H}$	
B	bh	$b\overline{h}$	b
$\overline{B}$	$\overline{b}h$	$\overline{bh}$	$\overline{b}$
	h	$\overline{h}$	N

bh é o número de exemplos para os quais B é verdade e H é verdade.
$b\overline{h}$ é o número de exemplos para os quais B é verdade e H é falso.
$\overline{b}h$ é o número de exemplos para os quais B é falso, mas H é verdade.
$\overline{bh}$ é o número de exemplos para os quais B é falso e H é falso.
b é o número total de exemplos para os quais B é verdade.
$\overline{b}$ é o número total de exemplos para os quais B é falso.
h é o número total de exemplos para os quais H é verdade.
$\overline{h}$ é o número total de exemplos para os quais H é falso.
N é o número total de exemplos.

Tabela 2.3: Matriz de contingência para uma regra R

	H	$\overline{H}$	
B	f_{bh}	$f_{b\overline{h}}$	f_b
$\overline{B}$	$f_{\overline{b}h}$	$f_{\overline{bh}}$	$f_{\overline{b}}$
	f_h	$f_{\overline{h}}$	1

Tabela 2.4: Matriz de contingência com freqüências relativas para uma regra R

Usando como base a matriz de contingência, é possível definir a maioria das medidas propostas na literatura para avaliação de regras.

2.7 Medidas de Avaliação de Regras

Várias medidas têm sido propostas na literatura com o objetivo de avaliar regras individuais de conhecimento. Dentre esses trabalhos, destaca-se o trabalho de Lavrač et al. (1999), no qual é proposto tratar esses medidas dentro de um mesmo *framework*, utilizando como base a matriz de contingência com freqüências relativas. Segue uma descrição dessas medidas dentro do *framework* proposto por Lavrač et al. (1999), utilizando as freqüências relativas — Tabela 2.4 — bem como as freqüências absolutas — Tabela 2.3.

Precisão (2.1): A precisão (consistência ou confidência) é uma medida do quanto uma regra é específica para o problema. A precisão pode ser definida como a probabilidade condicional de H ser verdade dado que B é verdade. Quanto maior, mais precisamente a regra cobre a classe em questão.

$$Acc(R) = P(H|B) = \frac{P(HB)}{P(B)} = \frac{f_{bh}}{f_b} = \frac{bh}{b} \tag{2.1}$$

Laplace (2.2): A medida de precisão não considera o número de exemplos cobertos corretamente, o que privilegia regras que cubram corretamente poucos exemplos. A correção de Laplace tenta contornar esse problema penalizando regras que cubram poucos exemplos. Nessa definição N_{Cl} é o número de classes possíveis.

$$LAcc(R) = \frac{f_{(bh+1)}}{f_{(b+N_{Cl})}} = \frac{bh+1}{b+N_{Cl}} \tag{2.2}$$

Erro (2.3): O erro de uma regra é definido como $1 - Acc(R)$. Quanto maior o erro, menos precisamente a regra cobre a classe em questão.

$$Err(R) = 1 - Acc(R) = P(\overline{H}|B) = \frac{f_{b\overline{h}}}{f_b} = \frac{b\overline{h}}{b} \tag{2.3}$$

Confiança Negativa (2.4): é o correspondente à precisão, mas para os exemplos que não são cobertos pela regra. É definida como a probabilidade condicional de H ser falso dado que B também é falso.

$$NegRel(R) = P(\overline{H}|\overline{B}) = \frac{P(\overline{HB})}{P(\overline{B})} = \frac{f_{\overline{bh}}}{f_{\overline{b}}} = \frac{\overline{bh}}{\overline{b}} \tag{2.4}$$

Sensibilidade (2.5): Sensibilidade (completeza ou *recall*) é uma medida do número (relativo) de exemplos da classe prevista em H cobertos pela regra. É definida como a probabilidade condicional de B ser verdade dado que H é verdade. Quanto maior a sensibilidade, mais exemplos são cobertos pela regra.

$$Sens(R) = P(B|H) = \frac{P(HB)}{P(H)} = \frac{f_{bh}}{f_h} = \frac{bh}{h} \tag{2.5}$$

Especificidade (2.6): é o correspondente à sensibilidade, mas para os exemplos que não são cobertos pela regra R. É definida como a probabilidade condicional de B ser falso dado que H é falso.

$$Spec(R) = P(\overline{B}|\overline{H}) = \frac{P(\overline{HB})}{P(\overline{H})} = \frac{f_{\overline{bh}}}{f_{\overline{h}}} = \frac{\overline{bh}}{\overline{h}} \tag{2.6}$$

Cobertura (2.7): Cobertura é uma medida do número (relativo) de exemplos cobertos pela regra R. É definida como a probabilidade de B ser verdade. Quanto maior a cobertura, maior o número de exemplos cobertos pela regra R.

$$Cov(R) = P(B) = f_b = \frac{b}{N} \tag{2.7}$$

Suporte (2.8): Suporte (freqüência) é uma medida do número (relativo) de exemplos cobertos corretamente pela regra R. É definido como a probabilidade de H e B serem verdade. Quanto maior o suporte, maior o número de exemplos da classe em questão que são cobertos corretamente pela regra R.

$$Sup(R) = P(HB) = f_{bh} = \frac{bh}{N} \tag{2.8}$$

Novidade (2.9): Novidade pode ser definida como a probabilidade de B e H ocorrerem juntos não puder ser inferida pelas probabilidades de B e H isoladamente, isto é, B e H não são estatisticamente independentes. A medida de novidade é obtida comparando o valor esperado $P(HB)$ com os valores de $P(H)$ e $P(B)$. Quanto mais o valor esperado diferir do observado, maior é a probabilidade que exista um correlação verdadeira e inesperada entre B e H. Pode ser demonstrado que $-0,25 \leq Nov(R) \leq 0,25$, e quanto maior um valor positivo (mais próximo de 0,25) mais forte é a associação entre B e H enquanto que, quanto maior um valor negativo (mais próximo de -0,25), mais forte é a associação entre B e $\overline{H}$.

$$Nov(R) = P(HB) - P(H)P(B) = f_{bh} - f_h \times f_b = \frac{1}{N} \times \left(bh - \frac{h \times b}{N} \right) \tag{2.9}$$

Satisfação (2.10): Satisfação é o aumento relativo na precisão entre a regra $B \rightarrow verdade$ e a regra $B \rightarrow H$. É uma medida mais indicada para tarefas voltadas à descoberta de conhecimento, sendo capaz de promover um equilíbrio entre regras com diferentes condições e conclusões.

$$Sat(R) = \frac{P(\overline{H}) - P(\overline{H}|B)}{P\overline{H}} = \frac{f_{\overline{h}} - \frac{f_{b\overline{h}}}{f_b}}{f_{\overline{h}}} = 1 - \frac{b\overline{h} \times N}{b \times \overline{h}} \tag{2.10}$$

Precisão Relativa (2.11): A precisão relativa de uma regra mede o ganho de precisão obtido em relação à precisão de uma regra padrão $verdade \rightarrow H$, ou seja, que avalia B como verdade para todos os exemplos. Nesse caso, uma regra só interessa se melhorar a precisão da regra padrão.

$$RAcc(R) = P(H|B) - P(H) = \frac{f_{bh}}{f_b} - f_h = \frac{bh}{b} - \frac{h}{N} \tag{2.11}$$

Confiança Negativa Relativa (2.12): É o análogo a precisão relativa para os exemplos que não são cobertos pela regra. Nesse caso, a regra padrão é $falso \rightarrow \overline{H}$.

$$RNegRel(R) = P(\overline{H}|\overline{B}) - P(\overline{H}) = \frac{f_{\overline{bh}}}{f_{\overline{b}}} - f_{\overline{h}} = \frac{\overline{bh}}{\overline{b}} - \frac{\overline{h}}{N} \tag{2.12}$$

Sensibilidade Relativa (2.13): A sensibilidade relativa mede o ganho de sensitividade obtido em relação à sensibilidade de uma regra padrão $B \rightarrow verdade$, ou seja, uma regra que avalia H como verdade para todos os exemplos.

$$RAcc(R) = P(B|H) - P(B) = \frac{f_{bh}}{f_h} - f_b = \frac{bh}{h} - \frac{b}{N} \tag{2.13}$$

Especificidade Relativa (2.14): É o análogo a sensibilidade relativa para os exemplos que não são cobertos pela regra. Nesse caso, a regra padrão é $\overline{B} \rightarrow falso$.

$$RSpec(R) = P(\overline{B}|\overline{H}) - P(\overline{B}) = \frac{f_{\overline{bh}}}{f_{\overline{h}}} - f_{\overline{b}} = \frac{\overline{bh}}{\overline{h}} - \frac{\overline{b}}{N} \tag{2.14}$$

Um ponto importante referente as medidas relativas (medidas 2.11, 2.12, 2.13 e 2.14) é que elas dão mais informação sobre a utilidade de uma regra que as informações fornecidas pelas suas respectivas medidas absolutas (medidas 2.1, 2.4, 2.5 e 2.6). Por exemplo, se a precisão de uma regra é menor que a freqüência relativa da classe que a regra prediz, então a regra tem um desempenho ruim, independentemente de sua precisão absoluta.

Existe, no entanto, um problema com a precisão relativa, pois é fácil obter uma alta precisão relativa para regras muito específicas, ou seja, regras com uma baixa generalidade de $P(B)$. Para contornar esse problema, é proposto em Lavrač et al. (1999) uma variante das medidas relativas, na qual é atribuído um peso para cada uma dessas medidas. Esse peso promove uma balanceamento entre a generalidade e a relatividade dessas medidas.

Dessa forma, a medida $RAcc(R)$ é multiplica pelo seu coeficiente de peso, $P(B)$, obtendo a nova medida **Precisão Relativa com Peso** (2.15), também conhecida na literatura como ganho. É possível mostrar que essa medida é equivalente a $Nov(R)$, ou seja, regras com alta precisão relativa também tem alta novidade e vice-versa.

$$WRAcc(R) = P(B)(P(H|B) - P(H)) = f_b\left(\frac{f_{bh}}{f_b} - f_h\right) \equiv Nov(R) \tag{2.15}$$

Analogamente, as medidas **Confiança Negativa** (2.16), **Sensibilidade** (2.17) e **Especificidade** (2.18) **Relativas com Peso** são obtidas multiplicando as correspondentes medidas relativas, $RNegRel(R)$, $RSens(R)$ e $RSpec(R)$, pelos coeficientes de pesos $P(\overline{B})$, $P(H)$ e $P(\overline{H})$, respectivamente. No trabalho de Lavrač et al. (1999) é demonstrado que essas medidas são equivalentes entre si (Equação 2.19), ressaltando a importância da precisão relativa com peso como medida fundamental para a avaliação de regras, promovendo um balanceamento entre precisão e as outras medidas.

$$WRNegRel(R) = P((\overline{B}))P(\overline{H}|\overline{B}) - P(\overline{H}) = f_{\overline{b}}\left(\frac{f_{\overline{bh}}}{f_{\overline{b}}} - f_{\overline{h}}\right) = \frac{\overline{b}}{N}\left(\frac{\overline{bh}}{\overline{b}} - \frac{\overline{h}}{N}\right) \tag{2.16}$$

$$WRSens(R) = P(H)(P(B|H) - P(B)) = f_h\left(\frac{f_{bh}}{f_h} - f_b\right) = \frac{h}{N}\left(\frac{bh}{h} - \frac{b}{N}\right) \tag{2.17}$$

$$WRSpec(R) = P((\overline{H}))P(\overline{B}|\overline{H}) - P(\overline{B}) = f_{\overline{h}}\left(\frac{f_{\overline{bh}}}{f_{\overline{h}}} - f_{\overline{b}}\right) = \frac{\overline{h}}{N}\left(\frac{\overline{bh}}{\overline{h}} - \frac{\overline{b}}{N}\right) \tag{2.18}$$

$$WRAcc(R) \equiv WRNegRel(R) \equiv WRSens(R) \equiv WRSpec(R) \tag{2.19}$$

Muitas outras medidas para avaliação de regras podem ser encontradas na literatura. Freitas (1998a,b) faz o uso de medidas objetivas para avaliar qualidade e surpresa de regras. Também, o número de nós (atributos) presentes em uma árvores ou o número de testes presentes no corpo de uma regra são usados, normalmente, como medida de compreensibilidade sintática de regras — Equação 2.20.

$$Compr(R) = 1 - \frac{| < complexo > |}{M} \tag{2.20}$$

onde $| < \textit{complexo} > |$ refere-se ao número de condições no corpo B da regra R e M é o número de atributos presentes no conjunto de exemplos. $Compr(R)$ é zero quando todos os atributos participam do corpo da regra (máxima complexidade) e atinge seu máximo valor quando somente um atributo participa do corpo da regra, ***i.e.*** $Compr(R) = 1 - \frac{1}{M}$.

É importante ressaltar que existe a possibilidade de combinar as medidas aqui apresentadas para auxiliar o usuário no entendimento e utilização do conhecimento adquirido por sistemas de aprendizado. Também vale lembrar que, mesmo que a precisão global do classificador não seja considerada boa, podem existir algumas regras boas em termos de qualidade, novidade, interessabilidade e outras. Esse aspecto é muito importante na área de descoberta de conhecimento de bases de dados (KDD) (Fayyad et al., 1996a,b; Rezende et al., 2003).

Outro trabalho relacionado à combinação de regras de conhecimento utilizando as medidas aqui descritas, foi desenvolvido por Bernardini (2006).

2.8 Considerações Finais

Neste capítulo foram apresentados alguns conceitos de aprendizado de máquina importantes para o entendimento deste trabalho. Dentre os conceitos apresentados vale ressaltar a linguagem de descrição do conhecimento no formato de regras e as medidas de avaliação de regras individuais, descritas em um mesmo *framework* que permite obtê-las a partir da matriz de contingência de cada regra, as quais são utilizadas neste trabalho para guiar o algoritmo evolutivo na construção de regras de conhecimento com propriedades específicas.

Capítulo

3

COMPUTAÇÃO EVOLUTIVA

> Como a seleção natural trabalha unicamente para e pelo bem de cada espécie, todos os talentos mentais e corpóreos tenderão a progredir na direção da perfeição.
> Charles Darwin (1809–1882), *Origem das Espécies*

3.1 Considerações Iniciais

MUITOS são os termos para designar os programas computacionais desenvolvidos sob o paradigma evolutivo. Esses termos foram aparecendo ao longo da linha histórica da Computação Evolutiva (CE) , quase que em paralelo, devido à núcleos de desenvolvimento distintos. O objetivo deste capítulo é apresentar o que motivou o aparecimento da computação evolutiva, uma breve conceituação a respeito da seleção natural e evolução, bem como traçar uma linha histórica sobre a computação evolutiva em termos de sua origem, funcionamento e aplicações. Entretanto, uma breve introdução sobre otimização faz-se necessária para contextualizar a computação evolutiva.

3.2 Visão Geral sobre Otimização

Os problemas do mundo real são representados no mundo computacional por meio de modelos que variam de equações matemáticas à estruturas bastante complexas. Após a representação desses problemas muitas tarefas podem ser desempenhadas, como simulações, visualizações, integrações com outros modelos, etc. Uma dessas tarefas consiste em otimizar soluções de problemas representados por esses modelos, dando assim origem a uma área que trata dos problemas de otimização. Muitos estudos foram feitos e diversos métodos de otimização foram desenvolvidos em virtude desses estudos. Existem basicamente três classes de métodos: 1) baseados em cálculo; 2) enumerativos; e, 3) estocásticos.

Os métodos de otimização baseados em cálculo têm sido exaustivamente estudados, os quais se subdividem em duas classes: indiretos e diretos. Métodos indiretos buscam em extremos locais pela solução do sistema de equações não-lineares ajustando o gradiente da função objetivo como zero (Goldberg, 1989). Dada uma função contínua e sem restrições — Figura 3.1 — a busca por um possível ponto de pico começa pela restrição da procura naqueles pontos cuja inclinação em todas as direções é negativa. Por outro lado, os métodos diretos buscam por ótimos locais movendo a solução corrente na direção do gradiente local. Essa é a noção de busca usada pelo método *hill-climbing*, o qual encontra o máximo local escalando na direção do maior pico.

Embora os métodos baseados em cálculo tenham sido aprimorados, eles apresentam alguns problemas. Primeiramente, ambos métodos, indiretos e diretos, procuram por pontos locais na vizinhança do ponto corrente. Se a busca iniciar na vizinhança do menor pico, considerando o exemplo ilustrado na Figura 3.1 , isso fará com que o ponto de pico máximo não seja atingido. Ainda, ambos métodos assumem que as funções são continuas e portanto admitem derivadas para a obtenção do gradiente. Entretanto, os espaços de busca de problemas do mundo real são compostos por funções descontínuas, multimodais e com ruido, tal como a representada na Figura 3.2 . Não há dúvidas que métodos que dependam da existência de continuidade e derivadas da função objetivo são aplicáveis apenas a domínios muito restritos (Haupt & Haupt, 1998).

Uma outra proposta consiste de esquemas enumerativos. A idéia é bastante simples: dentro de um espaço finito de busca, ou um espaço de busca discretizado e infinito, o algoritmo de busca inicia procurando por valores da função objetivo em todo o espaço de busca, um de cada vez. Embora esse esquema

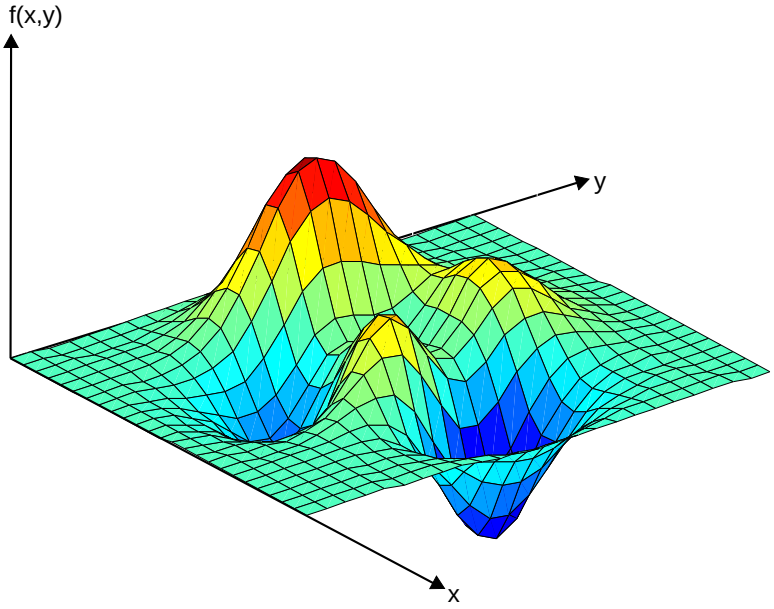

Figura 3.1: A função $f(x, y)$ possui pontos de mínimo e máximo obtidos por meio da combinação precisa dos parâmetros x e y.

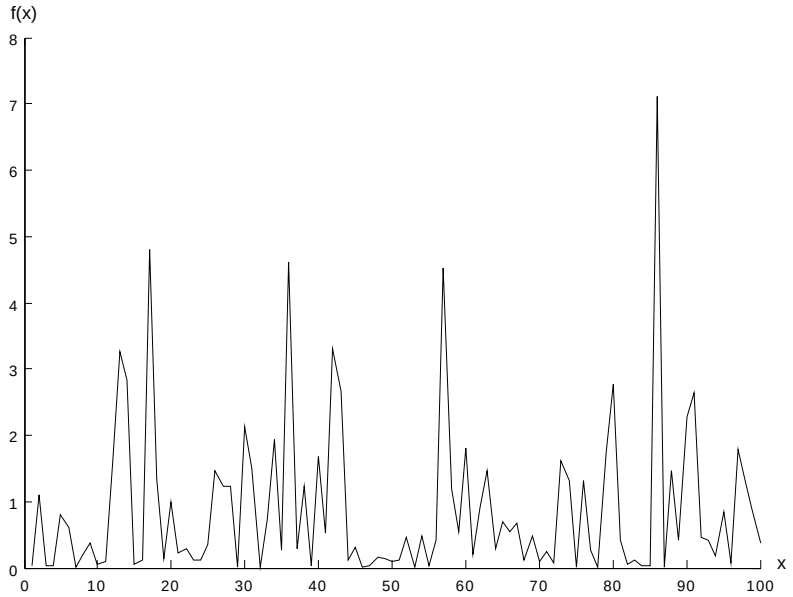

Figura 3.2: A função $f(x)$ possui pontos de mínimo e máximo que não podem ser encontrados por métodos que utilizem o gradiente da função.

de busca seja muito simples, seu comportamento na busca é semelhante a dos humanos quando o espaço de busca é pequeno. Entretanto, esse método torna-se extremamente ineficiente quando aplicado a espaços de busca

maiores.

A terceira classe de métodos de busca são os métodos estocásticos. Nessa classe os parâmetros da função assumem valores aleatórios que são armazenados e comparados uns com os outros em relação ao valor da função objetivo. Na realidade, os métodos de busca aleatórios possuem uma eficiência muito próxima à dos métodos enumerativos. Porém, deve-se fazer uma distinção entre os métodos aleatórios e os métodos estocásticos. Os algoritmos evolutivos são um exemplo desses últimos métodos, os quais utilizam escolhas aleatórias para guiar métodos altamente exploratórios (estocásticos).

Um outro método de busca bastante conhecido é o *simulated annealing*. O nome vem do processo de enrijecimento na metalurgia, que é uma técnica que envolve o aquecimento e o esfriamento controlado do material para a melhoria das suas propriedades. Nesse processo, o metal atinge sua configuração mais estável minimizando sua energia interna total. O principio do método *simulated annealing* é similar. Cada ponto no espaço de busca tem uma energia associada, que indica o quão bom aquele ponto é como solução do problema. O objetivo é encontrar o ponto com a menor energia (Koza, 1992). Um método de busca mais recente é aquele baseado em colônias de formigas, no qual há uma interação entre os membros da colônia como forma de cooperação, adaptação e solução de problemas (Parpinelli et al., 2002).

Os três métodos aqui citados — Algoritmos Evolutivos, *Simulated Annealing* e Algoritmos baseados em Colônias de Formigas — são métodos de busca que vem sendo amplamente empregados em espaços de busca complexos, nos quais outras técnicas mais específicas não são factíveis. O emprego desses métodos, com destaque para os algoritmos evolutivos, se deve ao fato de não ser necessária a modelagem de funções matemáticas que representem o problema e nem o cálculo de derivadas e gradientes. Esses três métodos foram criados com base na observação de fenômenos naturais que se mostram muito eficazes quando aplicados na natureza e, portanto, crê-se que o mesmo desempenho seja possível quando aplicados a problemas modelados computacionalmente (Mitchell, 1998).

Dessa forma, uma vez que existia uma lacuna entre os problemas do mundo real e os métodos de otimização baseados em cálculo de derivadas e gradiente, pensou-se em simular nos computadores o mecanismo evolutivo e de seleção que ocorre na natureza. Na próxima seção é dada uma visão geral sobre evolução e seleção natural.

3.3 Seleção Natural e Evolução

Nos anos que antecederam o século XIX, a única teoria aceita em relação às diferentes espécies animais e vegetais existentes no planeta é chamada Criacionismo. Segundo essa teoria, todas as espécies animais e vegetais são obras Divinas e suas características atuais são as mesmas daquelas iniciais quando Deus as criou. No entanto, esta teoria deixa uma lacuna em relação ao aparecimento de novas espécies e à extinção de outras. Para complementar o Criacionismo, surgiu uma outra teoria denominada Catastrofismo. De acordo com essa teoria, que procurava não entrar em atrito com a Bíblia, e ainda condizia com o Dilúvio, novas espécies surgiam e outras eram extintas devido à catástrofes naturais.

No início do século XIX surgiram muitas teorias que contradiziam as demais teorias aceitas até aquele momento. Uma das mais importantes foi aquela desenvolvida por Jean-Baptiste Lamarck (1744 - 1829), cujo pressuposto do surgimento de novas espécies estava relacionado à *"Lei do Uso e Desuso"*, segundo à qual, as características dos indivíduos eram melhoradas de acordo com seu uso repetitivo ou, ainda, essas características eram enfraquecidas, ou até mesmo removidas, por causa do desuso. Essas características modificadas eram então passadas para as gerações futuras dos descendentes desses indivíduos. Em 1830, o geólogo inglês Sir Charles Lyell refutou a Teoria do Catastrofismo, mas não fez o mesmo com a teoria de Lamarck, denominada *Lamarckismo*.

A grande modificação em relação às teorias aceitas e como novas espécies surgem e outras são extintas mudou completamente em 1859. Foi nesse ano que o cientista inglês Charles Darwin — Figura 3.3 — publicou sua obra mais famosa, intitulada *"Teoria da Evolução das Espécies"* (Darwin, 1859). Segundo sua teoria, todos os indivíduos de uma população são diferentes. Devido à essas diferenças, alguns deles são melhor adaptados ao ambiente, se comparados aos demais e, portanto, possuem melhores chances de sobreviver e procriar. Essas características vantajosas são herdadas pelos indivíduos das gerações descendentes, tornando-se através do tempo características predominantes na população. Essa teoria ficou conhecida como *Darwinismo*.

Seguindo os preceitos do Darwinismo, podemos definir o termo *evolução* como sendo as mudanças ocorridas em uma espécie ao longo do tempo devido a pressões do ambiente no qual essa espécie vive. Assim, as espécies sobreviventes são aquelas que ao longo do tempo evoluem satisfazendo as pressões do ambiente, enquanto que as espécies extintas são aquelas incapazes de satis-

Figura 3.3: Cientista inglês Charles Robert Darwin (1809–1882), criador da Teoria da Evolução das Espécies.

fazer essas pressões. À esse mecanismo evolutivo em função das pressões do ambiente da-se o nome de *seleção natural*. A seleção natural é o mecanismo essencial para a evolução cuja explicação foi proposta por Charles Darwin e, nos dias de hoje, ainda é aceita pela comunidade acadêmica como a melhor explanação para o aparecimento e extinção das espécies.

O conceito envolvido na seleção natural está no fato que as condições ambientais determinam como certas características particulares dos organismos servem para a sobrevivência e reprodução desses organismos. Organismos que não possuem características favoráveis em relação ao meio em que vivem podem não reproduzirem, ou até mesmo morrerem. Quando as condições ambientais permanecem as mesmas, ou similares o suficiente para serem favoráveis àquelas características, estas tendem a ser mais comuns dentro da população.

A teoria da evolução das espécies pela seleção natural, proposta por Darwin, parte da premissa que as características dos organismos variam de uma forma não-determinística dos pais para seus descendentes. Esse processo é chamado individualização. Embora a teoria de Darwin não explique como esse processo ocorre, descobertas científicas recentes no campo da genética mostram seu mecanismo de funcionamento (Gould, 2002).

O mecanismo da seleção natural não faz qualquer distinção relacionada às características serem favoráveis em relação ao ambiente ou em relação à aptidão reprodutiva. O mecanismo de seleção natural simplesmente considera que se uma variação particular nos organismos manifestar um melhor poder de sobrevivência ou reprodução, seus descendentes terão mais chances de sobreviver e procriar do que os demais organismos que não manifestaram aquela variação. As características originais irão desaparecer, assim como qualquer

variação que implique em mal adaptação. Então, certas características são preservadas devido às vantagens seletivas que elas provém aos indivíduos, permitindo assim que esses indivíduos produzam mais descendentes que os indivíduos que não as possuem. Eventualmente, após muitas gerações, os organismos irão desenvolver características adaptativas cada vez mais complexas.

O que torna uma característica mais propensa ao sucesso é altamente dependente dos fatores ambientais. Quando membros de uma espécie são separados eles enfrentam diferentes condições ambientais, tendendo a se desenvolverem em direções diferentes. Após um grande período, suas características terão se desenvolvido através de caminhos diferentes, e os indivíduos anteriormente separados não poderão mais se reproduzir. Nesse ponto eles são considerados espécies distintas. Esta é a razão pela qual uma espécie as vezes se separa em múltiplas espécies, ao invés de simplesmente ser substituída por uma nova espécie.

Conforme explanada no livro *Teoria da Evolução das Espécies* (Darwin, 1859), a seleção natural pode ser expressa como a seguir:

1.	**Se**	*Existem organismos que reproduzem,*	**e**
2.	**Se**	*Os descendentes herdam características de seus progenitores,*	**e**
3.	**Se**	*Existe variabilidade das características,*	**e**
4.	**Se**	*O ambiente não pode suportar todos os membros de uma população em crescimento*	
5.	**Então**	*Os membros da população com características menos adaptativas irão morrer,*	**e**
		Os membros com características mais adaptativas irão ter sucesso	

O resultado desse processo é a evolução das espécies. Note que este é um processo contínuo e considera como as espécies sofrem mudanças, podendo tanto levar à extinção de espécies quanto à criação de novas espécies. Note também que a lei acima não se aplica somente a organismos biológicos. Ela se aplica a todos os organismos que reproduzem numa forma que envolva herança e variabilidade (Darwin, 1859). Portanto, uma forma de seleção natural poderia ocorrer no mundo não-biológico — *i.e.*, computação evolutiva em geral. Neste trabalho especificamente, os algoritmos evolutivos serão a forma não-biológica de explorar os mecanismos da seleção natural e da evolução para a solução de problemas computacionais, especificamente, para a construção de regras simbólicas de conhecimento com propriedades específicas.

Na próxima seção são apresentados os primeiros relatos do uso desses con-

ceitos como forma de contribuir para a solução de problemas computacionais.

3.4 Áreas da Computação Evolutiva

Os primeiros relatos do uso de modelos computacionais para o melhor entendimento do processo natural da evolução começou na década de 50. As primeiras descrições do uso desses modelos aparecem nos artigos de Friedberg (1958) e Friedberg et al. (1959). Esses trabalhos representam os primeiros passos da computação evolutiva e aprendizado de máquina. Outro trabalho importante da época foi o desenvolvido por Fisher (1958), segundo o qual a evolução é uma forma de adaptação tal qual ocorre no aprendizado, diferindo tão somente quanto à escala de tempo. Ainda, houve o trabalho publicado por Bremermann (1962), que utilizou-se da simulação evolutiva como forma de otimização de problemas numéricos. Bremermann também desenvolveu alguns dos primeiros algoritmos evolutivos (Bremermann et al., 1965)

Entretanto, foi na metade da década de 60 que foi desenvolvida a base do que hoje se pode identificar como as três principais áreas dentro da computação evolutiva (De Jong et al., 2000). A base da Programação Genética foi desenvolvida por Fogel et al. (1966) em San Diego, Califórnia. O início dos Algoritmos Genéticos ocorreu na Universidade de Michigan em Ann Arbor por Holland (1967). Enquanto que, do outro lado do Atlântico, Estratégias Evolutivas foi um desenvolvimento conjunto de um grupo de três estudantes, Bienert, Rechenberg e Schwefel, em Berlin (Rechenberg, 1965).

Na Figura 3.4 são apresentadas as subáreas envolvidas na computação evolutiva. Essa é uma visão de como a computação evolutiva esta subdividida, a qual foi esquematizada seguindo descrições contidas em várias publicações que são referência para essa área (Goldberg, 1989; Michalewicz, 1997; Haupt & Haupt, 1998; Mitchell, 1998; Bäck et al., 2000; Heitkoetter & Beasley, 2001).

A seguir são brevemente descritas cada uma das sub-áreas que compõem a computação evolutiva.

3.4.1 Programação Evolutiva

Na sua forma padrão, um programa evolutivo utiliza-se das quatro características principais de um algoritmo evolutivo: 1) inicialização; 2) variação; 3) avaliação; e, 4) seleção.

Figura 3.4: A Computação Evolutiva pode ser vista como uma grande área que envolve todas as áreas cujo desenvolvimento está calcado no preceito da evolução.

Para um programa evolutivo, o aprendizado é decorrente de um processo evolutivo que retém os indivíduos de sucesso que são treinados por meio de um processo estocástico de tentativa e erro. A função de avaliação mede diretamente a aptidão, ou equivalentemente o erro comportamental, de cada membro da população com relação ao ambiente. A seleção retira probabilisticamente as soluções sub-ótimas da população (Porto, 2000).

Um algoritmo básico de programação evolutiva começa com uma população de soluções que são inicializadas de forma aleatória, heurística ou outro método apropriado. O tamanho da população pode variar, mas geralmente seu tamanho permanece constante ao longo do processo. Cada uma das soluções é avaliada com respeito a uma função de avaliação específica do problema. Após a criação da população de soluções iniciais, cada um dos indivíduos é alterado pela aplicação do processo de mutação. Uma característica forte que diferencia programação evolutiva das demais abordagens é a não adoção de um mecanismo de recombinação. Cada um dos indivíduos gera descendentes, os quais são replicados através de um mecanismo estocástico de erro (mutação).

Programação evolutiva difere das outras técnicas da computação evolutiva, tal como os algoritmos genéticos, de uma maneira crucial. Programação evolutiva é uma abordagem de otimização *top-down* ao invés de uma abordagem *bottom-up*. A idéia de que a soma de partes ótimas raramente leva a uma solução ótima geral é a chave da diferença nessa abordagem. Algoritmos genéticos

são baseados na identificação, combinação e sobrevivência dos melhores blocos de construção (esquema), os quais são iterativamente combinados para formar blocos de construção maiores e melhores. Na hipótese dos blocos de construção há uma premissa implícita que pressupõe a separação da função de aptidão ao longo do espaço de busca, tornando os algoritmos genéticos um método de otimização localmente focalizado. Já na programação evolutiva a abordagem é global. Em programação evolutiva, da mesma forma que em estratégias evolutivas, o operador de variação permite a mudança simultânea de todas as variáveis ao mesmo tempo. A aptidão, descrita em termos do comportamento de cada indivíduo da população, é avaliada indiretamente, sendo a base da sobrevivência de cada indivíduo (Porto, 2000).

Desde o aparecimento da programação evolutiva muito esforço vem sendo feito de forma a otimizar seus mecanismos. Em especial, nas décadas de 80 e 90 houve um grande avanço e interesse por parte da comunidade científica no que se refere aos algoritmos evolutivos. Na literatura encontram-se muitas aplicações da programação evolutiva, tais como treinamento, construção e otimização de redes neurais, otimização de rotas (em duas, três ou mais dimensões), projeto de novos medicamentos, automação de controles e teoria dos jogos. Notavelmente, aplicações de sucesso incluem problemas de ordem NP, descritos em Porto (2000).

3.4.2 Estratégias Evolutivas

Os esforços em minimizar o atrito total de corpos tridimensionais em fluxos turbulentos foi, e ainda é, o objetivo geral das pesquisas nos institutos de hidrodinâmica. Três estudantes — Peter Bienert, Ingo Rechenberg e Hans-Paul Schwefel — encontraram-se no Instituto Hermann Föttinger da Universidade Técnica de Berlin em 1964. Como eles estavam fascinados, não somente com aerodinâmica, mas também com a cibernética, eles decidiram solucionar o problema intratável de como desenvolver formas com o auxílio de algum tipo de robô. O robô deveria ser capaz de manipular um modelo flexível posicionado dentro de um túnel de vento. Esperava-se que ao final o robô fosse capaz de modelar o objeto de forma a obter um objeto delgado. Entretanto, o robô não conseguiu alcançar esse objetivo, porque o tratamento de uma variável de cada vez e os métodos baseados em gradientes acabavam por levar a pontos de mínimo local (Rudolph, 2000). Ingo Rechenberg teve a idéia de introduzir pequenas mudanças aleatórias que só eram permitidas se essas mudanças levassem à melhorias no modelo. Essa idéia foi o passo inicial para o nascimento das Estratégias Evolutivas (Rechenberg, 1965).

Essa abordagem, da forma como foi criada, não animou a comunidade acadêmica da época devido a sua estratégia aleatória. Mais tarde, a idéia foi estendida para a aplicação em problemas multi-dimensionais e, ainda, foi incluída a operação de seleção. Muitas outras melhorias foram feitas, tais como a implementação de operações de cooperação e deterioração (Schwefel & Rudolph, 1995). Na verdade, embora as estratégias evolutivas pareçam uma abordagem aleatória, o operador de mutação — principal guia de exploração do espaço de busca nessa abordagem — é fortemente baseado em probabilidades, sendo que a viabilidade desse operador pode ser demonstrada algebricamente (Rudolph, 2000).

3.4.3 Algoritmos Genéticos

O uso dos métodos nos moldes que os levaram a ser chamados de algoritmo genético surgiram na década de 60. Esses métodos foram desenvolvidos por John Holland e seu grupo de estudos na Universidade de Michigan e sua publicação *"Adaptação em Sistemas Naturais e Artificiais"* (Holland, 1975), lhe rendeu o título de criador dos Algoritmos Genéticos. Entretanto, o uso de seu trabalho se deve em grande parte a um de seus alunos, David Goldberg e sua publicação intitulada *"Algoritmos Genéticos em Busca, Otimização e Aprendizado de Máquina"* (Goldberg, 1989).

Embora os méritos da criação dos algoritmos genéticos se devam à Holland, outros trabalhos anteriores à sua publicação já mostravam sinais da utilização de algumas de suas idéias. A primeira referência ao termo *algoritmo genético* surgiu com um trabalho desenvolvido por Bagley (1967). A simulação de células biológicas foi estudada por Rosenberg (1967). Esse trabalho foi importante para o posterior desenvolvimento da teoria de algoritmos genéticos aplicados a sistemas artificiais (Goldberg, 1989). Uma aplicação aproximada de algoritmos genéticos foi realizada por Cavicchio (1970) no problema de reconhecimento de padrões. A primeira aplicação de algoritmos genéticos a um problema de otimização matemática foi desenvolvida por Hollstein (1971). A codificação dos indivíduos utilizando números reais é resultado do trabalho publicado por Bosworth et al. (1972). Um marco importante no desenvolvimento dos algoritmos genéticos é devido ao trabalho de De Jong (1975). Nessa publicação, foi combinada a teoria dos esquemas proposta por Holland com os resultados dos trabalhos realizados por De Jong. Esse estudo é considerado de grande importância ao desenvolvimento dos algoritmos genéticos e possui diversas conclusões cuidadosamente formuladas (Haupt & Haupt, 1998).

Embora os algoritmos genéticos sejam utilizados com representações de seqüen-

cias de *bits*, outras representações foram propostas, tal como os parâmetros com valores reais. Além disso, novas formas de *crossover* e mutação, além de outros operadores, vêm sendo introduzidos gerando variações de algoritmos genéticos (Eshelman, 2000). Contudo, há quem defenda que aplicações baseadas em algoritmos genéticos devam estar representadas como seqüências de *bits* e utilizar somente os operadores clássicos dessa abordagem (Michalewicz, 1997).

Na literatura, a programação genética e os sistemas de classificação, descritos brevemente a seguir, são tratados como subáreas dos algoritmos genéticos, tal como representado na Figura 3.4 na página 27.

Programação Genética é uma forma de algoritmo evolutivo, distinguindo-se pelo uso de um conjunto particular na escolha da representação, dos operadores genéticos e da função de avaliação. A programação genética é implementada como um algoritmo evolutivo no qual as estruturas de dados que sofrem adaptação são na verdade programas computacionais. A função de avaliação na programação genética envolve a execução desses programas (Kinnear Jr., 2000). Portanto, a programação genética utiliza uma busca direcionada pela evolução, sendo que o espaço de busca é composto por programas que solucionam determinado problema, entre os quais existe um possível programa que produz o melhor resultado.

Na programação genética, para criar uma população inicial de soluções, um grande número de programas é gerado aleatoriamente. Cada um desses programas é executado e o resultado da execução é utilizado para compor o valor da avaliação. Então, uma nova população de programas é criada por meio da cópia de alguns programas selecionados. Essa população é complementada com a criação de novos programas que são gerados pela aplicação dos operadores genéticos em programas já existentes e previamente selecionados com base em sua aptidão. Novamente, todos os indivíduos são avaliados e novos valores de aptidão são atribuídos. Esse ciclo é repetido até que o critério de parada seja satisfeito.

Em alto nível, programação genética é definida como um algoritmo genético com algumas escolhas diferenciadas feitas na representação do problema, nos operadores genéticos e na função de avaliação empregados. Na programação genética, diferentemente das demais abordagens, a forma como os indivíduos são representados para um problema em particular pode ser vista como uma linguagem. Essa visão na forma de representação pode ser útil na modelagem dos operadores e da função de avaliação (Kinnear Jr., 2000).

Aprendizado de Sistema de Classificação é normalmente referenciado como sendo a primeira técnica de aprendizado de máquina que utiliza algoritmos genéticos (Mitchell & Taylor, 1999). Também, é descrita como uma técnica de aprendizado de máquina que utiliza algoritmos genéticos como meio principal para a descoberta de conhecimento. Entretanto, os detalhes de operação de um sistema de classificação variam muito entre a vasta gama de implementações e pode ser considerado mais um conceito do que um algoritmo (Smith, 2000).

3.5 Algoritmo Evolutivo

Algoritmos baseados em computação evolutiva utilizam as noções da seleção natural para simular um método de busca que implementa uma seleção artificial, na qual os indivíduos de melhor aptidão têm maior tendência à sobreviver e propagar suas características. Para tanto, é necessário um formalismo de representação e funcionamento que simule computacionalmente o que ocorre na natureza, os operadores evolutivos e de seleção, bem como a forma de calcular a aptidão dos indivíduos. Como a forma mais conhecida de algoritmo evolutivo é um algoritmo genético, nesta seção, em alguns casos, para fins de simplicidade e para melhor ilustrar os conceitos, será adotada a representação binária.

Diferentes problemas requerem diferentes representações para o uso da computação evolutiva. Entretanto, na maior parte dos casos a representação é semelhante. Assume-se a existência de uma população inicial de indivíduos $P(t)$ ($t = 0$) que, geralmente, são inicializados de forma aleatória[1]. Cada indivíduo é uma possível solução para o problema tratado, sendo que a população inicial será formada por possíveis soluções para esse problema. Dada a população inicial de indivíduos, cada um desses indivíduos será avaliado pela função de avaliação[2] com relação à real possibilidade de solucionar adequadamente o problema. Então, aos indivíduos são atribuídos valores que refletem de forma probabilística suas chances de serem escolhidos para a reprodução e criação de novos indivíduos. Quanto maior for o valor recebido na avaliação, maior será a probabilidade de seleção para a reprodução. Assim, um número de

[1]Neste trabalho, por exemplo, a população inicial consiste de regras de conhecimento quaisquer relacionadas ao mesmo conjunto de dados.

[2]Os termos *função de avaliação* e *função de aptidão* se referem à função que avalia a aptidão do indivíduo quanto à solução de um determinado problema e, portanto, são termos que possuem significado idênticos. Na maior parte dos casos, a *função objetivo* que é a função a ser otimizada é utilizada como função de avaliação. Nesses casos, os três termos possuem significados idênticos.

indivíduos é selecionado aleatoriamente, respeitando suas probabilidades de seleção. Os indivíduos são então, aos pares, geneticamente modificados (por meio dos operadores evolutivos) e novos indivíduos são gerados (indivíduos filhos), dando origem a uma nova população $P(t)$. Esse ciclo de seleção, recombinação e avaliação continua ($t = 1, 2, ..., n$) até que o melhor indivíduo seja encontrado ou algum outro critério de parada seja satisfeito. Um Algoritmo Evolutivo — AE — típico é representado pelo Algoritmo 3.1 (Mitchell, 1998).

Algoritmo 3.1 Algoritmo Evolutivo

```
1: t ⟵ 0
2: inicializar P(0)
3: avaliar P(0)
4: enquanto critério de parada não satisfeito faça
5:    t ⟵ t + 1
6:    selecionar P(t) de P(t − 1)
7:    modificar P(t)
8:    avaliar P(t)
9: fim-enquanto
```

Cada ciclo de um algoritmo evolutivo corresponde à criação de novos indivíduos a partir dos indivíduos já existentes, utilizando para tanto os operadores evolutivos. Então, os novos indivíduos mais aptos tendem a substituir os indivíduos anteriores e menos aptos. Esse ciclo descrito corresponde a uma geração. Tal como descrito por Darwin, o que ocorre é que os indivíduos mais aptos tendem a sobreviver, enquanto que os indivíduos menos aptos ao meio tendem a desaparecer.

3.5.1 Representação dos Indivíduos

Como dito anteriormente, num algoritmo evolutivo a população é formada por um conjunto de indivíduos que são possíveis soluções para o problema. Existem muitas formas de representar os indivíduos, a qual dependerá muito do problema sendo solucionado. Em geral, os indivíduos são representados por uma seqüencia de *bits* que codificam uma solução. Assim, a população é composta por várias seqüências de *bits* representadas separadamente. O tamanho do indivíduo depende diretamente do número de parâmetros levado em consideração e da variabilidade que esses parâmetros assumem. Quanto maior o número de parâmetros e da variabilidade dos valores dos parâmetros, maior será o tamanho da seqüência de *bits*. Fazendo analogia com a área biológica, a seqüência de *bits* é chamada de cromossomo, por conter as características de cada indivíduo. Na verdade, os indivíduos são bastante simplificados, de forma que cada cromossomo corresponde a um indivíduo (Goldberg, 1989). Entretanto, uma representação binária não supre as reais necessidades do

trabalho aqui proposto. Como o nosso objetivo é utilizar os algoritmos evolutivos como ferramenta na construção de regras de conhecimento com propriedades específicas, existem duas representações mais apropriadas para definir os indivíduos. São as abordagens *Michigan* e *Pittsburgh*, descritas a seguir.

Representação Michigan: Proveniente dos trabalhos de Holland e Reitman na década de 70 na Universidade de Michigan, originou-se a forma de representação de regras que ficou conhecida como *Michigan* em referência ao nome da universidade de origem.

Nessa abordagem, cada indivíduo da população representa uma única regra, *i.e.* uma parte da solução candidata composta por todas as regras (população) (Michalewicz, 1997). Na abordagem Michigan, após vários ciclos a solução é dada, em geral, por uma única regra evoluída que representa o melhor indivíduo da população. Entretanto, nem sempre uma única regra é capaz de representar toda a solução de um problema. Existem formas de co-evoluir os indivíduos de forma que toda a população faça parte da solução (Smith, 2000). Ainda, outra forma é repetir sucessivas vezes o ciclo evolutivo do algoritmo genético de forma a obter-se várias regras. Entretanto, esse é um processo lento que demanda um grande esforço computacional.

Já no contexto deste trabalho, a representação Michigan possui a grande vantagem de permitir a atuação direta do algoritmo evolutivo em cada regra de conhecimento. Com isso, cada regra da população pode vir a desenvolver características específicas e desejadas.

Representação Pittsburgh: Os trabalhos desenvolvidos por De Jong e Smith na Universidade de Pittsburgh deram origem à forma de representação de regras, a qual ficou conhecida como *Pittsburgh*, em referência ao nome da universidade de origem (Smith, 2000).

Diferentemente da abordagem Michigan, nessa abordagem cada indivíduo da população representa um conjunto de regras candidatas à solução do problema. Dessa forma, a população contém vários conjuntos de regras, sendo que cada indivíduo (conjunto de regras) representa uma solução geral do problema. Comparativamente com a abordagem Michigan, a abordagem Pittsburgh requer um esforço computacional menor para obter a solução, embora o cálculo da aptidão dos indivíduos seja mais complexa que na outra abordagem (Freitas, 2002). Em geral, utiliza-se essa abordagem quando o objetivo é evoluir um conjunto de regras para obter-se um classificador.

Existe uma vantagem nessa representação, conforme explicado por Frei-

tas (2002). Em um bom classificador, é desejável que as regras que o compõem cubram de forma satisfatória uma porção significativa do conjunto de dados. Ao mesmo tempo, deseja-se que essas regras não contenham muita informação redundante. A sobreposição de regras é freqüentemente indesejável, pois produz classificadores menos compreensíveis — no sentido de que o classificador é constituído por um conjunto maior de regras que cobrem exemplos comuns. Na Figura 3.5 é ilustrada a sobreposição de regras. As regras de conhecimento R1, R2, R3 e R4 cobrem uma grande quantidade de exemplos comuns.

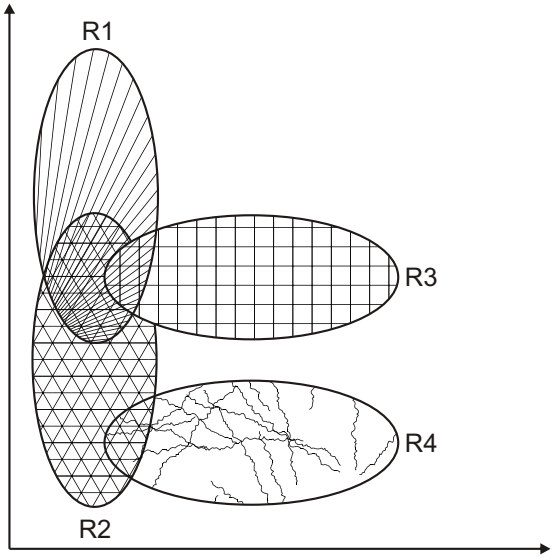

Figura 3.5: Exemplo de sobreposição de regras

A construção de classificadores com esse tipo de regras pode ser contornada com a representação Pittsburgh. Como cada indivíduo representa um conjunto de regras de conhecimento, é possível moldar uma função de avaliação que considere a relação entre essas regra. Dessa forma, indivíduos que representem classificadores com menor sobreposição podem ser considerados superiores aos indivíduos que possuem sobreposição muito elevada. Na representação Michigan, entretanto, a função de avaliação analisa cada indivíduo desconsiderando o relacionamento do mesmo com o resto da população, o que exige a aplicação de outros mecanismos para gerar classificadores constituídos de regras que cobrem exemplos diferentes.

Deve ser observado que, no contexto deste trabalho, a representação Pittsburgh possui um contraponto significativo. Uma vez que cada indivíduo representa um conjunto de regras, é difícil projetar operadores evolutivos capazes de considerar regras que possam ser evoluídas indi-

vidualmente. Por esse motivo, é adotada neste trabalho a representação Michingan.

3.5.2 Seleção

Um dos passos mais importantes em um algoritmo evolutivo é o processo de *seleção*. Nesse processo, os indivíduos são selecionados para a geração da próxima população e a forma como esse processo ocorre pode ser determinante para o sucesso ou fracasso do algoritmo evolutivo. No processo de seleção existe um coeficiente que mede a *pressão da seleção*[3] perante os indivíduos da população (Haupt & Haupt, 1998). Quanto maior a pressão da seleção, mais rapidamente o algoritmo converge para uma solução, a qual não é necessariamente a melhor. Isso ocorre porque a velocidade da convergência é afetada pela pressão da seleção, fazendo com que nem todo o espaço de busca seja adequadamente vasculhado. Por outro lado, uma baixa pressão de seleção faz com que a convergência seja lenta e leve muito tempo para encontrar uma solução adequada, embora seja feita uma varredura mais ampla no espaço de busca.

Existem muitos métodos de seleção, sendo que neste trabalho são descritos a seguir aqueles mais utilizados pela comunidade acadêmica.

3.5.2.1 Roda da Roleta

No método de seleção roda da roleta, cada indivíduo recebe uma probabilidade de ser sorteado, a qual é calculada pela proporção da aptidão do indivíduo com o total da aptidão acumulada. Dessa forma, a indivíduos de maior aptidão será atribuída uma probabilidade maior, como ilustrado na Figura 3.6 , tal que indivíduos com maior probabilidade possuam maiores chances de serem sorteados. Nessa figura tem-se cinco indivíduos (I_1, I_2, I_3, I_4 e I_5) dispostos de forma a representar suas probabilidades de sorteio quando a roleta girar e o indicador apontar qual é o indivíduo sorteado. Esse método de seleção pressupõe a reposição dos elementos, ou seja, quando um indivíduo é sorteado ele volta a figurar na lista dos possíveis indivíduos a serem sorteados.

Contudo, esse método de seleção apresenta alguns problemas quando a distribuição das probabilidades de sorteio tendem para extremos — um indivíduo possuí aptidão destacada em relação aos demais ou quando todos os indivíduos da população possuem aptidões muito próximas (Deb, 2000). Quando um indivíduo possui uma alta aptidão em relação aos demais indivíduos da

[3] A pressão de seleção é relativa a como o algoritmo evolutivo percorre o espaço de busca.

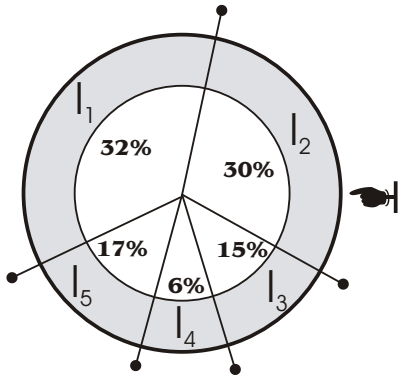

Figura 3.6: Método de seleção roda da roleta

população, a proporção da aptidão desse indivíduo em relação à aptidão acumulada tende a ser grande, assim como a probabilidade de ser o escolhido. Como os demais indivíduos da população possuem uma aptidão bem menor, se comparada ao *super indivíduo*, a tendência é que esse indivíduo seja selecionado muitas vezes, fazendo com que suas características se propaguem rapidamente pela população e causando uma convergência prematura do algoritmo evolutivo sem que, necessariamente, a solução ótima seja encontrada. Por outro lado, se todos os indivíduos da população apresentam aptidões muito próximas, a probabilidade de cada indivíduo ser sorteado também é aproximadamente a mesma. Logo, a população seguinte será formada, praticamente, pelos mesmo indivíduos da população anterior e a busca, geração após geração, tende a ter um comportamento aleatório (Grefenstette, 2000a).

3.5.2.2 Torneio

A seleção por torneio é um método muito simples que inicia pela escolha de um número fixo de indivíduos selecionados aleatoriamente da população atual. Então, dentre esses indivíduos, apenas o indivíduo que possuí a maior aptidão é copiado para a população seguinte. O processo se repete o número de vezes que for necessário até que a nova população esteja completa (Haupt & Haupt, 1998).

Esse método de seleção pode ser com ou sem reposição dos indivíduos já sorteados. Embora esse método de seleção pareça demandar bastante tempo computacional, sua ordem de complexidade é linearmente proporcional ao tamanho da população, pois, diferentemente do método de seleção roda da roleta, este método independe de uma ordenação prévia dos elementos e do cálculo das probabilidades de seleção (Blickle, 2000).

3.5.2.3 Ranking

O método de seleção por *ranking* não faz uso de probabilidades proporcionais à aptidão tal como no método de seleção roda da roleta, mas, dependendo do tipo de problema, os indivíduos são ordenados de forma decrescente ou crescente de acordo com os valores obtidos da função de avaliação. Então, para cada indivíduo é atribuído um número inteiro que representa o *ranking* em relação aos demais indivíduos da população. Quanto melhor o *ranking* do indivíduo, melhor sua aptidão em relação ao problema e, portanto, maiores suas chances de sorteio (Freitas, 2002; Grefenstette, 2000b). Neste trabalho foram utilizados dois tipos de *ranking* para a seleção dos indivíduos: linear e exponencial.

O *ranking* linear de um indivíduo I_i é dado pela Equação 3.1. Nessa equação, i é a ordem no *rank* decrescente da função de aptidão, de forma que aos indivíduos de melhor aptidão são atribuídos os menores *ranks*, e min e max são parâmetros de entrada para o cálculo do *rank* final, enquanto que $size$ é dado pelo número total de indivíduos da população. Quanto maior a diferença entre min e max maior será a pressão de seleção e, portanto, a chance de convergência prematura.

$$LinearRank(I_i) = min + (max - min) \times \left(\frac{size - i}{size - 1}\right) \tag{3.1}$$

Já o *ranking* exponencial de um indivíduo I_i é dado pela Equação 3.2. Como no caso anterior, i é a ordem no *rank* decrescente da função de aptidão e $base \in (0..1)$ é um parâmetro de entrada. Quanto menor o valor de $base$ maior será a pressão de seleção e, portanto, a chance de convergência prematura.

$$ExponentialRank(I_i) = base^{i-1} \tag{3.2}$$

3.5.2.4 Elitismo

Nos métodos de seleção para a composição dos indivíduos da próxima população é comum a cópia do melhor indivíduo da população atual para a nova população. Esse artifício é chamado de elitismo, e tem por objetivo privilegiar a melhor solução atual de forma a aumentar as chances desse indivíduo propagar as suas características entre os demais indivíduos da população. Também, ao indivíduo selecionado por elitismo, evita-se a aplicação dos operadores de *crossover* e mutação que podem degenerar a solução representada por aquele indivíduo.

3.5.3 Operadores

Como os algoritmos evolutivos são um método de busca baseados na evolução das melhores soluções, os operadores que guiam a busca adotam a mesma filosofia. Existem dois operadores que influenciam na busca por novas soluções: *crossover* e mutação.

3.5.3.1 Crossover

O operador de *crossover* tem o objetivo de gerar dois novos indivíduos a partir de outros dois indivíduos previamente selecionados. Os algoritmos evolutivos partem do pressuposto que a junção das melhores partes de determinadas soluções devem originar a melhor solução. Partindo dessa premissa, o operador de *crossover* combina os melhores indivíduos originando dois novos indivíduos que, a princípio, espera-se que sejam melhores que os indivíduos pais. O que o operador de *crossover* faz é guiar a busca no sentido de encontrar a melhor solução, de forma a recombinar as melhores soluções parciais e encontrar a solução ótima (Carvalho et al., 2003).

Existem vários tipos de operadores de *crossover*. O mais simples é o *crossover* de um ponto, no qual dados dois indivíduos a serem recombinados, um ponto é aleatoriamente escolhido. Assim, determina-se onde os cromossomos serão divididos e compartilharam suas partes. Na Figura 3.7 é mostrado o funcionamento do *crossover* de um ponto.

Figura 3.7: Representação do operador de crossover de um ponto. Dois novos indivíduos são gerados trocando-se as partes destacadas pelos retângulos de traços contínuo e pontilhados

Outra variação desse mesmo operador é o *crossover* de dois pontos, no qual a sistemática é semelhante, mas dois pontos são aleatoriamente selecionados. Nessa variação, obviamente, a recombinação é maior. Na Figura 3.8 é mostrado o funcionamento do *crossover* de dois pontos.

Partindo desses dois tipos de *crossover*, pode-se generalizar a idéia para o *crossover* de c-pontos. Quanto maior o c utilizado na definição do *crossover*, maior será o efeito da recombinação na geração dos novos filhos. Entretanto, isso afeta diretamente os esquemas sendo construídos e, portanto, o algoritmo

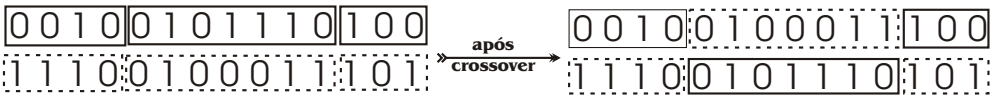

Figura 3.8: Representação do operador de crossover de dois pontos. Dois novos indivíduos são gerados trocando-se as partes destacadas pelos retângulos de traços contínuo e pontilhados

pode demorar, ou talvez nunca convergir para a solução esperada.

A possibilidade dos indivíduos sofrerem *crossover* é determinada pela *taxa de crossover*. Essa taxa informa a probabilidade de ocorrer *crossover* nos pares de indivíduos a cada novo emparelhamento. Normalmente, a taxa de *crossover* é igual para todos os indivíduos durante todas as gerações.

3.5.3.2 Mutação

Diferentemente do operador de *crossover*, o operador de mutação atua *in loco*, não gerando portanto um novo indivíduo. Se por um lado o *crossover* leva em consideração a hipótese dos blocos de construção e guia a busca na direção do melhor indivíduo, a mutação faz exatamente o oposto. Com base na *taxa de mutação*, que normalmente é um valor de baixa magnitude (0,1% a 0,5%), a cada nova geração a probabilidade de cada novo indivíduo sofrer mutação é considerada segundo essa taxa. Se o indivíduo for escolhido para sofrer mutação, então um *bit* é aleatoriamente escolhido, o qual é então invertido. Na Figura 3.9 é representado o operador de mutação.

Figura 3.9: Representação do operador de mutação. Alteração genética *in loco* que inverte o *bit* selecionado

O papel desse operador é inserir diversidade na população, fazendo com que os indivíduos explorem novas áreas do espaço de busca (Michalewicz, 1997). Deve ser observado que o operador de mutação é muito importante para evitar mínimos e máximos locais. Entretanto, não se deve considerar altas taxas de mutação, pois a inserção de muita diversidade na população pode fazer com que ela não convirja para uma solução ótima, fazendo com que a convergência da população oscile indefinidamente.

3.5.4 Função de Avaliação

A cada nova geração, novos indivíduos são gerados utilizando os operadores de *crossover* e mutação, os quais passam a figurar entre os indivíduos já existentes. Entretanto, é necessário uma forma eficaz de qualificar cada indivíduo como sendo apto ou não à solução do problema. Para isso, os algoritmos evolutivos utilizam a *função de avaliação* para medir a aptidão de cada individuo. Como a aptidão para a solução do problema é fortemente ligada à função sendo otimizada, normalmente é possível utilizar a função de avaliação como sendo a própria função que define o problema (função objetivo), tal que, $avaliar(P(t)) = F(P(t))$ (Freitas, 2002). Em outros casos ela não são idênticas. Por exemplo, se a função f a ser otimizada fosse dada pela Equação 3.3.

$$f(x) = x^2 \times \sin(x),\ -2 \leq x \leq 2 \tag{3.3}$$

Nesse caso, como a função $f(x)$ assume o valor máximo quando $x = 2$, basta utilizar a função de avaliação como sendo o próprio x.

Quando o problema envolve a otimização de um único objetivo, deseja-se encontrar a melhor solução disponível, chamada de ótimo global, ou ao menos uma boa aproximação dessa solução. Problemas com somente um objetivo são chamados de simples-objetivo. Contudo, são freqüentes os casos em que não existe somente um objetivo a ser otimizado, mas vários objetivos que na maior parte dos casos são conflitantes (Coello et al., 2002). Esses problemas com duas ou mais funções objetivo são chamados multi-objetivo e requerem a utilização de algoritmos computacionais e métodos matemáticos diferentes daqueles utilizados para solucionar problemas com um único objetivo. Na verdade, a noção de ótimo é diferente quando considera-se problemas multi-objetivo porque é desejável um equilíbrio entre todos os objetivos ao invés de somente um objetivo atingir o ponto ótimo (Coello, 2006).

Neste trabalho, o objetivo é construir regras de conhecimento com propriedades específicas, as quais estão relacionadas às medidas de avaliação de regras definidas na Seção 2.7 na página 13. Quando somente uma dessas medidas é otimizada, tem-se um problema simples-objetivo, enquanto que otimizar um conjunto dessas medidas constitui um problema multi-objetivo. Neste trabalho propomos uma solução simples para tratar esse último problema utilizando *ranks*.

3.6 Considerações Finais

Uma visão geral das sub-áreas que compõem a computação evolutiva foi apresentada neste capítulo. Essas sub-áreas estão calcadas sobre o funcionamento do mecanismo evolutivo que ocorre na natureza. O aparecimento da computação evolutiva foi motivado, entre outros, pela impossibilidade de aplicação dos métodos de otimização numérica para alguns problemas do mundo real.

Com a convergência das técnicas, a literatura aponta para uma convergência entre as denominações, mais especificamente nos casos em que o uso de várias técnicas evolutivas se faz presente (Bäck et al., 2000; Deb, 2001; Freitas, 2002; Coello et al., 2002). Dessa forma, decidimos enquadrar este trabalho como um algoritmo evolutivo, para o qual foi apresentado uma visão mais detalhada de funcionamento e possíveis representações de indivíduos, por utilizar uma representação própria para a codificação dos indivíduos e operadores evolutivos específicos para atuar sobre essa representação.

Uma característica importante de um algoritmo evolutivo é a codificação (representação) do indivíduos. No caso específico deste trabalho que tem como objetivo construir regras de conhecimento com propriedades específicas, representar regras de conhecimento como indivíduos de um algoritmo evolutivo é ponto chave na proposta. Dessa forma, no próximo capítulo é apresentada a proposta de aplicação de um algoritmo evolutivo em aprendizado de regras de conhecimento com propriedades específicas.

Capítulo

4

ALGORITMO EVOLUTIVO PROPOSTO

Darwin insistia que sua teoria não explicava somente a complexidade do corpo do animal, mas o complexidade de sua mente.
S. Pinker (1954), *Como a Mente Funciona*

4.1 Considerações Iniciais

CONSTRUIR regras de conhecimento com propriedades específicas, utilizando um algoritmo evolutivo, demanda a especificação de um formalismo de representação de regras de conhecimento sob a ótica desse paradigma. Além disso, os operadores evolutivos de *crossover* e mutação necessitam ser compatíveis com a estrutura de representação de indivíduos (regras) para guiar o processo evolutivo. A função de avaliação também precisa ser específica para o problema da construção de regras, uma vez que os indivíduos da população representam regras simbólicas. Neste capítulo é apresentada a estrutura proposta para a codificação de regras de conhecimento enquanto indivíduos da população do algoritmo evolutivo. Essa estrutura permite definir uma vasta gama de operadores evolutivos de *crosso-*

ver e mutação para atuarem sobre ela. A estrutura proposta também permite que qualquer das medidas de avaliação de regras apresentadas na Seção 2.7 seja facilmente calculada. Essas medidas podem tanto ser utilizadas individualmente como função de aptidão, como poder ser consideradas em uma função de aptidão multi-objetivo baseada em *rankings*, a qual considera de forma concomitante mais de uma medida de avaliação de regras, transformando-as numa função simples-objetivo. O critério de convergência do algoritmo evolutivo proposto também é apresentado neste capítulo, o qual considera a estabilidade da melhor solução encontrada ao longo das gerações do algoritmo evolutivo.

4.2 Solução Geral Proposta

A proposta deste trabalho consiste de um algoritmo evolutivo para a construção de regras de conhecimento com propriedades específicas. Ou seja, a população inicial do algoritmo evolutivo deve consistir de um conjunto inicial de regras de conhecimento relacionadas à um conjunto de dados, as quais devem estar representadas na estrutura específica que o algoritmo evolutivo reconhece para, assim, dar início ao processo. Portanto, é inicialmente necessário:

1. O conjunto inicial de regras de conhecimento relacionadas à um conjunto de dados.
2. Transformar essas regras para a estrutura de indivíduos do algoritmo evolutivo.

Na Figura 4.1 é ilustrada a idéia geral proposta neste trabalho: dado um conjunto de regras de conhecimento relacionadas à um conjunto de dados rotulados, as quais podem ser regras induzidas por algoritmos de aprendizado simbólico, algoritmos de geração de regras ou regras fornecidas pelo próprio usuário, o primeiro passo consiste em transformar essas regras iniciais, construídas de diversas formas, para um formato padrão de regras intermediário, o qual é posteriormente transformado para a estrutura de indivíduos do algoritmo evolutivo.

Neste trabalho propomos transformar essas regras, ou um subconjunto delas, para o formato padrão de regras $\mathcal{PBM}$[1] proposto por Prati et al. (2001), descrito a seguir, o qual permite, entre outros, unificar a linguagem de descrição

[1] A sigla $\mathcal{PBM}$ é em alusão aos sobrenomes dos idealizadores: Prati, Baranauskas e Monard.

de regras de conhecimento utilizadas por diversos algoritmos de aprendizado simbólico. Após a linguagem de descrição das regras estar unificada no formato $\mathcal{PBM}$, tem-se uma base de regras que pode ser utilizada parcialmente ou em sua totalidade. As regras escolhidas podem então ser transformadas para a estrutura de indivíduos do algoritmo evolutivo, inicializando assim a população inicial, sobre a qual métodos de seleção, operadores evolutivos e funções de avaliação, irão guiar uma tentativa de construir regras de conhecimento com as propriedades especificadas pelo usuário.

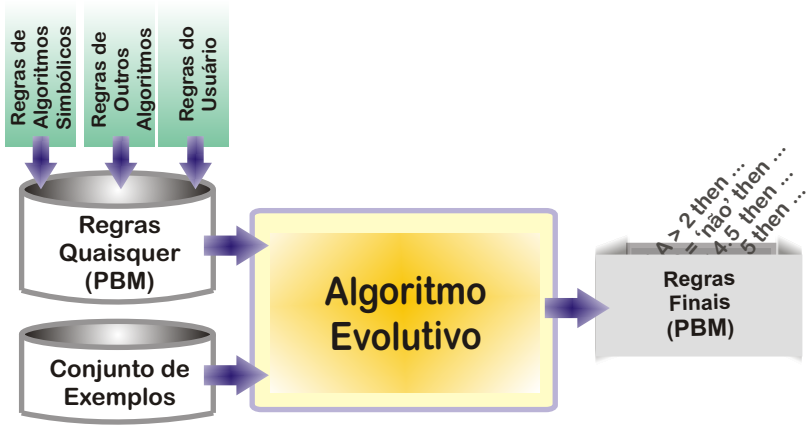

Figura 4.1: Solução geral proposta

4.3 Formato Padrão de Regras $\mathcal{PBM}$

Como mencionado, cada indivíduo do algoritmo evolutivo deve representar uma regra de conhecimento, a qual deve ser avaliada segundo uma medida ou um conjunto de medidas de avaliação de regras. Essas regras de conhecimento iniciais podem ser provenientes das mais diversas fontes, tais como de algoritmos de aprendizado, outros algoritmos e/ou definidas pelo usuário. A única restrição é que essas regras se refiram a um conjunto de exemplos de um mesmo domínio $\mathcal{D}$. Neste trabalho, adotamos o formato padrão de regras $\mathcal{PBM}$ como formato intermediário de regras a ser posteriormente transformado para a estrutura de indivíduos do algoritmo evolutivo descrita na Seção 4.4, construindo assim a população inicial do algoritmo evolutivo.

O formato padrão de regras $\mathcal{PBM}$ foi proposto por Prati et al. (2001), com o objetivo de unificar as diversas formas de representação de conhecimento na qual são expressos os classificadores induzidos por diferentes algoritmos de aprendizado simbólico. Além disso, foi também implementada por Prati uma

biblioteca de ferramentas que calcula informações relacionadas à matriz de contingência de cada regra, de forma padronizada, para um conjunto de algoritmos de aprendizado simbólico freqüentemente utilizados pela comunidade, a qual permite, facilmente, a inclusão de novos indutores. Essa biblioteca encontra-se integrada a um sistema computacional de maior porte — DISCOVER — que possui diversas outras facilidades utilizadas no desenvolvimento deste trabalho, como descrito na Seção 5.3.

O formato padrão de regras $\mathcal{PBM}$ estende a representação do formato padrão descrito na Seção 2.4, com a informação das freqüências relativas f_{bh}, $f_{b\overline{h}}$, $f_{\overline{b}\overline{h}}$ e $f_{\overline{b}h}$, além do número n de exemplos utilizados para calcular essas informações. Com essa informação, as freqüências marginais f_b, $f_{\overline{b}}$, f_h e $f_{\overline{h}}$ podem ser facilmente obtidas pela soma das linhas e colunas da matriz de contingência — Tabela 2.4 na página 13. Os valores absolutos das medidas também podem ser facilmente obtidos multiplicando as freqüências relativas por n. Assim, o formato $\mathcal{PBM}$ consiste da regra mais duas listas de elementos numéricos:

if <complexo> ***then*** <Class> $= C_i$ [f_{bh}, $f_{b\overline{h}}$, $f_{\overline{b}\overline{h}}$, $f_{\overline{b}h}$, n_k] [$f^?_{bh}$, $f^?_{b\overline{h}}$, $f^?_{\overline{b}\overline{h}}$, $f^?_{\overline{b}h}$, n_{uk}]

com $n_k + n_{uk} = n$, ***i.e.***, o número total de exemplos utilizado para calcular essas informações. Os elementos da primeira lista referem-se à matriz de contingência tal que todos os atributos que pertençam ao corpo da regra têm valores especificados em n_k dos exemplos, enquanto que a segunda lista considera aqueles exemplos nos quais pelo menos um dos atributos que participa do corpo da regra não está especificado em n_{uk} dos exemplos.

Dada uma regra de conhecimento no formato $\mathcal{PBM}$, ela deve ser posteriormente descrita na estrutura de representação de indivíduos utilizada pelo algoritmo evolutivo. A estrutura por nós proposta é descrita a seguir.

4.4 Representação dos Indivíduos

A representação e codificação dos indivíduos no algoritmo evolutivo constitui passo fundamental na busca por uma solução (Michalewicz, 1997). Isso se deve ao fato da representação dos indivíduos conter a maior parte das informações necessárias para a atuação dos operadores evolutivos e da função de avaliação. Isso leva à proposta de uma estrutura de dados adequada para representar cada indivíduo enquanto regra de conhecimento simbólica. Na Figura 4.2 encontra-se a estrutura da representação proposta para as regras enquanto indivíduos da população do algoritmo evolutivo. Cada regra

é formada pelo <complexo> com todas as possíveis condições (*condição*$_1$, ..., *condição*$_M$), onde M é o número de atributos do conjunto de exemplos ao qual as regras se referem, e pela decisão que aquela regra prediz. Nessa estrutura, as condições são formadas por limitantes inferiores e superiores. Em ambos limitantes existe uma tripla, sendo que:

- *bit* informa se o limitante está presente;
- *op* é o operador, sendo que $op \in \{<, \leq, >, \geq, =\}$;
- *vlr* é o valor que a condição assume para aquele limitante.

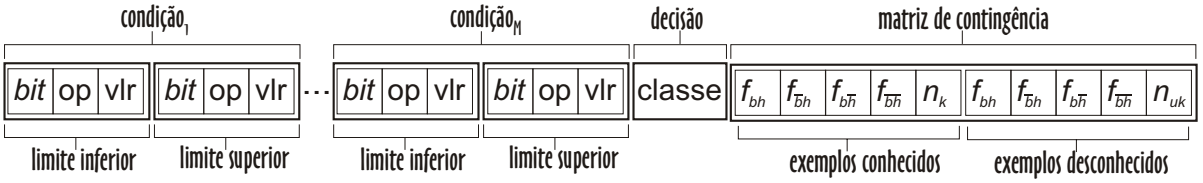

Figura 4.2: Representação de um indivíduo (regra)

Ainda, para cada regra encontra-se representada a matriz de contingência que contempla tanto a cobertura de exemplos com valores conhecidos (n_k) como de exemplos com valores desconhecidos (n_{uk}), calculada sobre um conjunto de $n = n_k + n_{uk}$ exemplos.

O tamanho de cada indivíduo é fixo, sendo que cada *condição*$_i$ está relacionada com o atributo X_i da tabela atributo-valor do conjunto de exemplos. Assim, o espaço necessário para representar um indivíduo é $O(M)$. Se a população inicial consiste de r regras de conhecimento, então o espaço necessário para representar a população é $O(r \times M)$. Caso o atributo X_i não seja considerado pela regra de conhecimento, tanto o *bit* do limitante inferior quanto o superior da *condição*$_i$ é zero. Deve ser observado que uma estrutura similar foi proposta por Freitas (2002). Entretanto, a codificação da regra com todos os limitantes, *i.e.* tamanho fixo, bem como a inicialização desses limitantes, como explicada a seguir, e com a matriz de contingência, é uma proposta deste trabalho.

Na estrutura ilustrada na Figura 4.2 os limitantes sempre possuem valores de inicialização obtidos do conjunto de dados fornecido e relacionado às regras. Dessa forma, a estrutura é mantida com tamanho fixo e o algoritmo evolutivo não corre o risco de gerar condições inválidas para as condições do complexo, um dos problemas apresentados por outras estruturas propostas para representar regras de conhecimento (Freitas, 2002). Como se trata de

uma inicialização dos valores, o *bit* de presença é sempre inicializado com zero. Essa inicialização dos valores depende do tipo do atributo que origina a condição da regra, sendo dividido em dois casos:

Qualitativo: Para os atributos qualitativos utiliza-se somente o limitante inferior para a codificação e o limitante superior é ignorado. Assim, o operador op é sempre o símbolo de igual ($=$) e o valor vlr assume a moda dos valores do atributo que origina aquela condição, no conjunto de dados considerado.

Quantitativo: Para os atributos quantitativos, para o limitante inferior o operador é o símbolo de maior-igual ($\geq$) e o valor vlr é o menor valor encontrado no conjunto de dados para aquele atributo. Para o limitante superior o operador é o símbolo menor-igual ($\leq$) e o valor vlr é o maior valor encontrado no conjunto de dados para aquele atributo.

Em ambos os casos, a decisão tomada na inicialização dos valores está relacionada à manutenção da cobertura por aquela regra representada por aquele indivíduo, deixando que a especialização ocorra quando da aplicação dos operadores evolutivos (*crossover* e mutação). Esses valores de inicialização (bit, op e vlr) são sobrescritos quando a população inicial é povoada com o conjunto inicial de regras já no formato $\mathcal{PBM}$. Dai, para cada regra, as condições sobrescrevem os valores de inicialização, tornando 1 (um) determinados *bits* de presença. Também, op e vlr são sobrescritos pelos pares presentes nas condições. No caso em que o complexo não tenha uma condição para sobrescrever os valores de inicialização, estes últimos permanecem intactos (*bit*=0) até a ocorrência de algum operador evolutivo.

Como exemplo desse processo de inicialização dos indivíduos que compõem a população do algoritmo evolutivo, considere o conjunto de dados da Tabela 4.1, o qual contém 3 (três) atributos e 20 (vinte) exemplos rotulados, relacionados à predição de concessão de empréstimo. Assim, as condições de uma regra qualquer podem ser compostas pelos atributos salário, casado e saldo e a classe é o atributo empréstimo. Cada condição é composta por um par de limitantes, os quais têm os *bits* que indicam a presença inicializados com zero. Na condição salário, para o limitante inferior o valor utilizado na inicialização é *min*(salário), enquanto que para o limitante superior o valor utilizado na inicialização é *max*(salário). De forma análoga o mesmo é feito para a condição que representa o saldo. No caso da condição que representa o atributo casado, por ser qualitativa, somente é utilizado o limitante inferior, já que não existe uma relação intervalar, mas somente o uso do operador de igualdade ($=$). Nesse caso, o valor de inicialização é a moda (valor mais freqüentemente

encontrado no conjunto de dados) do atributo casado. Os valores da classe e da matriz de contingência são determinados posteriormente.

	Salário (R$\times 10^3$)	Casado	Saldo (R$\times 10^3$)	Empréstimo
E_1	10	sim	1	não
E_2	9,5	não	6	sim
E_3	4	sim	1	sim
E_4	2	sim	3	sim
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
E_{20}	3	sim	4	sim

Tabela 4.1: Exemplo de dados no formato atributo valor

Na Figura 4.3 é exemplificada essa inicialização de condições, todas com *bit*=0. Nesse caso, utilizado o conjunto de dados da Tabela 4.1, foi encontrado que *max*(salário)=10; *min*(salário)=2; *moda*(casado)=sim; *max*(saldo)=6 e *min*(saldo)=0. Eles constituem os valores *default* das regras (indivíduos) iniciais, os quais serão sobrescritos com as verdadeiras condições das regras fornecidas, no formato $\mathcal{PBM}$, para participar da população inicial.

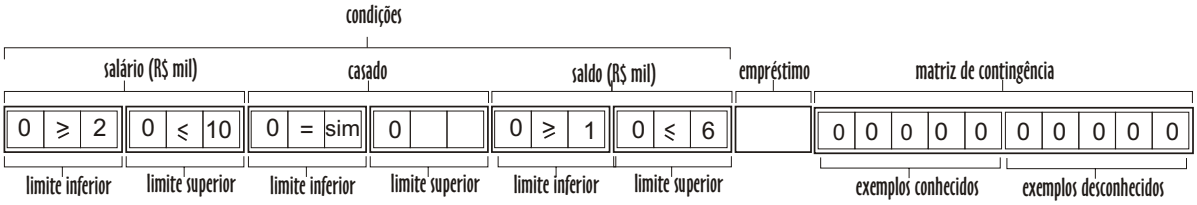

Figura 4.3: Exemplo de indivíduo com valores *default*

Por exemplo, a Regra 4.1 no formato $\mathcal{PBM}$ que expressa: "toda pessoa que possui salário entre R$ 2.000,00 e R$ 4.500,00 e é casado, é passível de receber um empréstimo", será codificada pelo indivíduo mostrado na Figura 4.4.

IF	**salário** ≥ 2	f_{bh}, $f_{b\overline{h}}$, $f_{\overline{bh}}$, $f_{\overline{b}h}$, n
	AND **salário** $\leq$ 4,5	[0.8000, 0.0000, 0.0000, 0.2000, 20]
	AND **casado** = sim	$f^{?}_{bh}$, $f^{?}_{b\overline{h}}$, $f^{?}_{\overline{bh}}$, $f^{?}_{\overline{b}h}$, n
THEN	**empréstimo** = *sim*	[0.0000, 0.0000, 0.0000, 0.0000, 0]

Regra 4.1: Exemplo de regra no formato $\mathcal{PBM}$

Deve ser notado que para as condições em que a regra $\mathcal{PBM}$ não fornece valores, foram mantidos os valores de inicialização e os *bits* de presença como zero. Ainda, essa regra cobre 20 exemplos — Tabela 4.1 — todos com valores conhecidos para os atributos $(n_k = 20)$, sendo que desse total 16 são cobertos

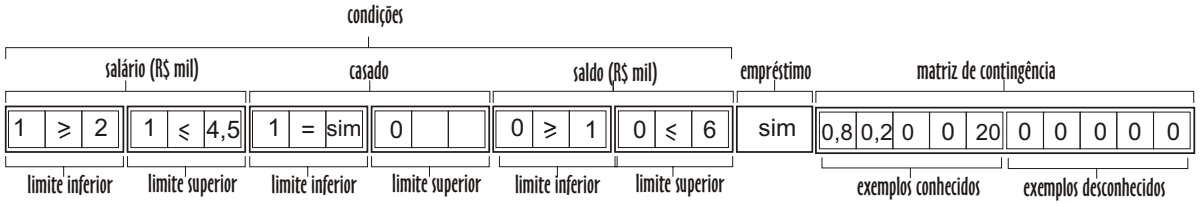

Figura 4.4: Regra 4.1 representada na estrutura de dados que codifica um indivíduo

corretamente ($f_{bh} = \frac{16}{20} = 0.8$) e 4 são cobertos incorretamente ($f_{b\overline{h}} = \frac{4}{20} = 0.2$). Os valores de $f^{?}_{bh}$, $f^{?}_{\overline{b}h}$, $f^{?}_{b\overline{h}}$ e $f^{?}_{\overline{b}\overline{h}}$ relacionados à n_{uk} para essa regra são zero, pois não há exemplos com valores desconhecidos para os atributos salário e casado que participam da regra ($n_{uk} = 0$). Em outras palavras, essa regra prediz corretamente a classe de 16 exemplos e prediz incorretamente a classe de 4 exemplos.

O fato do indivíduo ser codificado com tamanho fixo na população é importante porque não há a necessidade de fazer o alinhamento dos indivíduos para a atuação do operador de *crossover* (Freitas, 2002). Se os indivíduos fossem fiéis às regras $\mathcal{PBM}$ no que se refere ao tamanho e disposição das condições, *i.e.* tamanho variável, seria introduzido um problema para os operadores de *crossover*, já que para uma mesma condição válida sua posição em relação às demais seria diferente. Embora utilizando tamanho variável do indivíduo o espaço requerido para representar a população inicial seja apenas aquele ocupado para representar as condições presentes no complexo, o problema de alinhamento demandaria um elevado esforço computacional. Além disso, o processo de inicialização por nós proposto, facilita a implementação do operador de mutação, como será visto adiante.

Como os algoritmos evolutivos trabalham com a premissa da existência de partes boas em alguns indivíduos, as quais podem ser combinadas para gerar um indivíduo melhor (abordagem *bottom-up*) (Goldberg, 1989), existe a necessidade de não permitir que indivíduos potencialmente bons sejam desqualificados e banidos da população. Como também existe a preocupação de não deixar que essas condições *default* interfiram no processo evolutivo, foi decidido que esses valores *default* fossem os mais gerais possível. Dai a escolha da moda para atributos qualitativos e os limitantes inferior e superior contendo o *min* e o *max*, respectivamente, para atributos quantitativos.

4.5 Seleção

Os métodos de seleção desempenham papel fundamental na exploração do espaço de busca composto, neste trabalho, por regras de conhecimento. Como os métodos de seleção dependem apenas da função de avaliação para seu funcionamento, não há necessidade de reformulação dos métodos de seleção apresentados na Seção 3.5.2 na página 35, os quais foram considerados neste trabalho.

Deve ser feita uma consideração no caso do método de seleção por elitismo. No elitismo, o melhor indivíduo de cada geração é copiado para a população seguinte como forma de manter a melhor solução nas gerações seguintes. Entretanto, pode ser que em uma mesma população exista mais que um indivíduo (regra) que possua o melhor valor da função de avaliação naquela população. Dessa forma, deve ser escolhido um critério de desempate que resulte em apenas um entre os melhores indivíduos daquela população. Neste trabalho, esse indivíduo é escolhido aleatoriamente entre os indivíduos empatados, mas outros critérios de desempate podem ser considerados.

4.6 Operadores

Os operadores representam um papel fundamental nos algoritmos evolutivos por serem os agentes diretos na combinação dos indivíduos escolhidos a cada passo na busca de indivíduos ótimos. Os operadores descritos na Seção 3.5.3 na página 38 devem ser reformulados para a representação de indivíduos proposta neste trabalho, pois os operadores evolutivos clássicos não possuem funcionalidades adequadas para atuar sobre uma estrutura que não seja aquela binária. Na reformulação, descrita a seguir, há uma ampla gama de possibilidades, considerando a representação simbólica dos indivíduos.

4.6.1 *Crossover*

Foram considerados os seguintes três operadores de crossover, os quais serão melhor explicados na seções seguintes.

***Crossover* Estrutural:** No *crossover* estrutural, os indivíduos sofrem alteração na estrutura que os define. Assim, as regras de conhecimento representadas nos indivíduos podem ter suas condições recombinadas com

outros indivíduos.

***Crossover* em nível de Atributo:** No *crossover* em nível de atributo, os indivíduos sofrem alteração nos limitantes que definem as condições. Dessa forma, dados dois indivíduos deve ser feita uma inversão entre todo o limitante inferior (ou superior) que define a condição.

***Crossover* Local:** No *crossover* local, os indivíduos sofrem alteração nos valores que definem os limitantes. Dessa forma, dados dois indivíduos, os valores que definem os limitantes devem ser recombinados para gerar dois novos valores. Neste tipo de *crossover*, sua atuação depende do tipo de atributos que compõe a condição.

4.6.1.1 Crossover *Estrutural*

Neste tipo de *crossover* os indivíduos filhos são originados dos indivíduos pais que foram cruzados e tiveram seu material genético trocado no nível das condições. Dessa forma, pares inteiros de limitantes inferiores e superiores são trocados entre os pais para originarem os indivíduos filhos. Na Figura 4.5 é ilustrado o *crossover* estrutural[2]. Os indivíduos pais — Figura 4.5 **(a)** — que representam, respectivamente, as regras de conhecimento Regra 4.2 e Regra 4.3, são alinhados e o ponto de cruzamento aleatoriamente escolhido faz

IF **salário** ≥ 2
AND **salário** $\leq 4{,}5$
AND **casado** = sim
THEN **empréstimo** = *sim*

Regra 4.2: Regra Pai-1

IF **salário** ≥ 6
AND **saldo** $\geq 2{,}5$
THEN **empréstimo** = *sim*

Regra 4.3: Regra Pai-2

com que os indivíduos filhos — Figura 4.5 **(b)** — sejam gerados de forma que o primeiro indivíduo filho contenha a primeira porção do indivíduo pai inferior com a segunda porção do indivíduo pai superior. O segundo filho é obtido de maneira análoga.Os filhos gerados e representados na Figura 4.5 **(b)**, representam, respectivamente, as regras de conhecimento Regra 4.4 e Regra 4.5 no formato $\mathcal{PBM}$.

[2]Observar que as matrizes de contingência foram omitidas nessas figuras ilustrativas e nas regras de conhecimento para fins de simplificação.

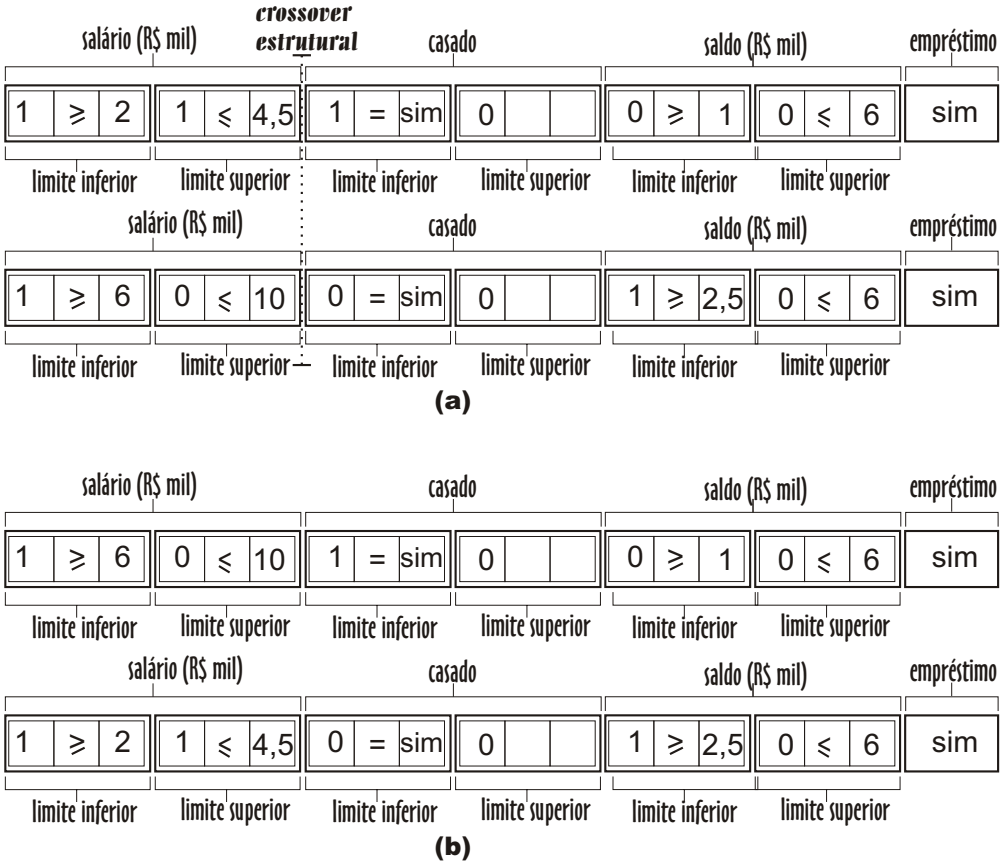

Figura 4.5: Operador de cruzamento em nível estrutural

IF **salário** ≥ 6
AND **casado** = sim
THEN **empréstimo** = *sim*

Regra 4.4: Regra Filho-1

IF **salário** ≥ 2
AND **salário** ≤ 4
AND **saldo** $\geq 2{,}5$
THEN **empréstimo** = *sim*

Regra 4.5: Regra Filho-2

4.6.1.2 Crossover em Nível de Atributo

Neste tipo de *crossover*, os indivíduos filhos são originados dos indivíduos pais que foram cruzados e tiveram seu material genético trocado no nível dos atributos. Dessa forma, limitantes inferiores e superiores são trocados entre os pais para originarem os indivíduos filhos. Na Figura 4.6 é ilustrado esse processo. Os indivíduos pais que representam as Regras 4.2 na página 52 e 4.3 na página 52 — Figura 4.6 **(a)** — são alinhados e o ponto de cruzamento aleatoriamente escolhido faz com que os indivíduos filhos — Figura 4.6 **(b)** — sejam gerados de forma que o primeiro indivíduo filho contenha a primeira porção do atributo salário do indivíduo que representa o segundo pai com a segunda porção do atributo salário do indivíduo que representa o segundo pai. O segundo filho é obtido de maneira análoga. Nesse processo, os demais atributos permanecem inalterados.

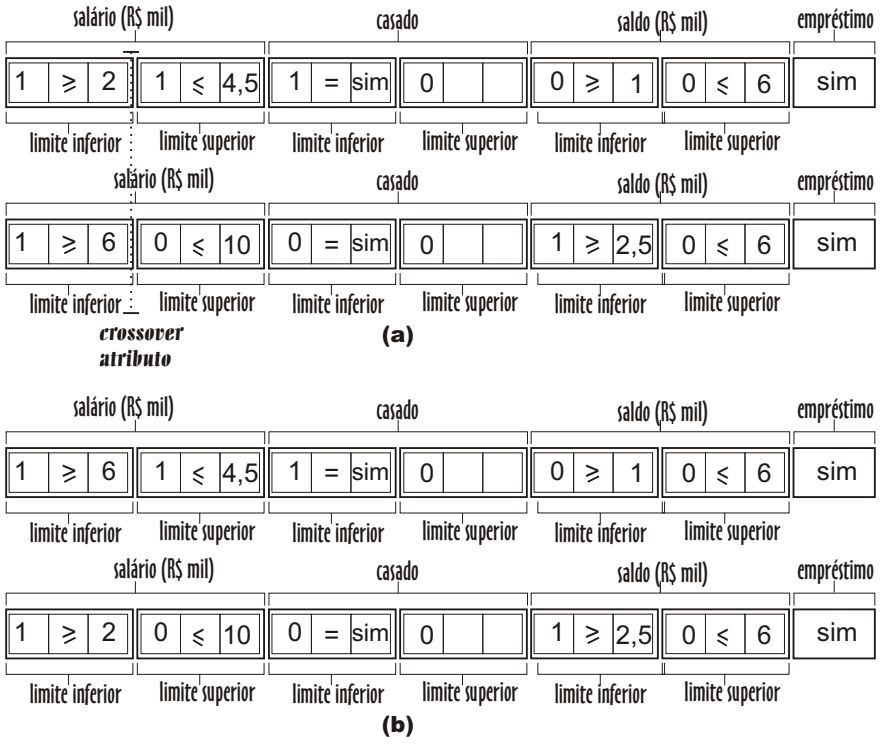

Figura 4.6: Operador de cruzamento em nível de atributo

Como pode ser observado no exemplo considerado para atributos quantitativos, o *crossover* em nível de atributo pode, eventualmente, gerar filhos que não cobrem nenhum exemplo. Esse é o caso do primeiro filho gerado, no qual o valor do salário deve ser "≥ 6" mas também "$\leq 4,5$", uma condição nunca atingida. Nesses casos o *crossover* é desfeito. Isso não acontece caso o *crossover* seja realizado com atributos qualitativos.

4.6.1.3 Crossover *Local*

Neste tipo de *crossover* os indivíduos filhos são originados dos indivíduos pais que foram cruzados e tiveram seu material genético trocado no nível dos valores locais que compõem as condições. Dessa forma, somente os valores de algum dos limitantes é recombinado. Na Figura 4.7 é ilustrado esse processo. Os indivíduos pais — Figura 4.7 **(a)** — são alinhados e o ponto de cruzamento aleatoriamente escolhido faz com que os indivíduos filhos — Figura 4.7 **(b)** — sejam gerados de forma que para o limitante selecionado ambos filhos contenham uma mistura do valor inicial. Na verdade, essa mistura pode ser uma simples inversão dos valores se os atributos envolvidos forem qualitativos. No caso do atributo ser quantitativo, utilizamos o procedimento descrito a seguir.

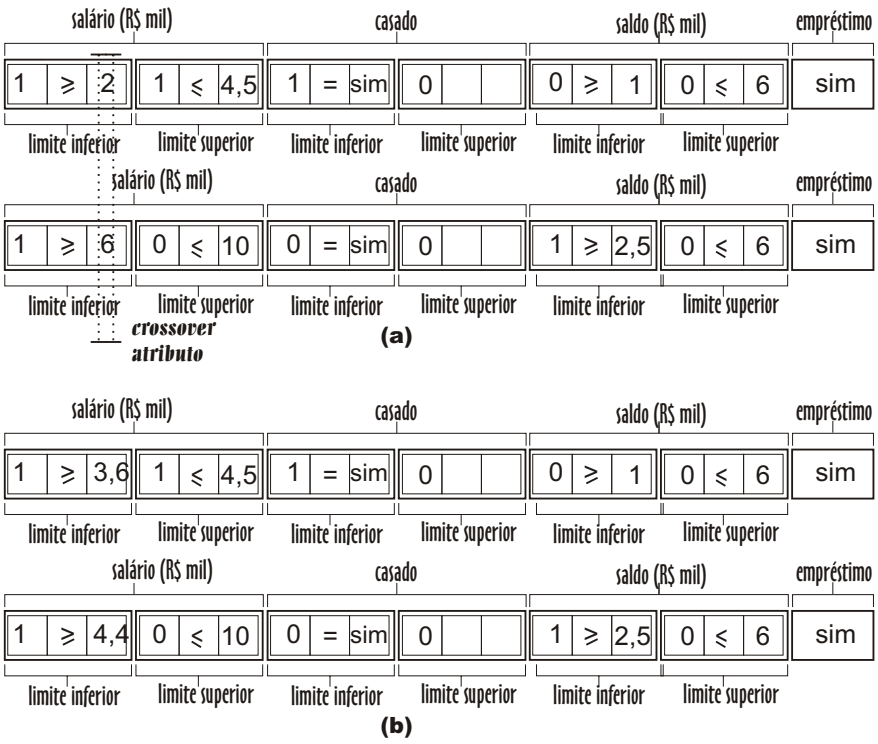

Figura 4.7: Operador de cruzamento em nível local

O *crossover* de nível local, para atributos quantitativos, considerado neste trabalho é o *BLX-*α (Herrera et al., 2002). Nesse *crossover*, dados dois valores reais iniciais (x e y), dois novos valores (x' e y') são gerados pela mistura (*blend*) que depende do fator α de mistura (com $0 \leq \alpha \leq 1$) — Equação 4.1.

$$\begin{cases} x' &= x - \alpha \times (x - y) \\ y' &= y + \alpha \times (x - y) \end{cases} \tag{4.1}$$

Como pode ser observado, os novos valores x' e y' são calculados utilizando uma fração da diferença existente entre os valores iniciais. No exemplo ilustrado na Figura 4.7 foi utilizado $\alpha = 0.4$, ou seja:

$$\begin{cases} x' &= 2 - 0.4 \times (2 - 6) = 3.6 \\ y' &= 6 + 0.4 \times (2 - 6) = 4.4 \end{cases} \tag{4.2}$$

Como explicado previamente, para o caso de atributos quantitativos, se o *crossover* local produzir uma condição que nunca é atingida, ele é desfeito.

Como os diferentes tipos de operadores de *crossover* atuam em partes distintas da estrutura que codifica o indivíduo (regra), consideramos interessante permitir que esses diferentes tipos de *crossover* atuem de forma concomitante em uma mesma população. Também, nos exemplos de *crossover* apresentados, somente pais que predizem a mesma classe foram escolhidos, mas nada impede que sejam utilizados pais que predizem classes diferentes.

4.6.2 Mutação

Foram considerados os seguintes dois tipos de mutação, os quais são explicados nas seções seguintes.

Mutação Estrutural: Na mutação estrutural, o indivíduo sofre alteração na estrutura que o representa. Assim, a regra de conhecimento representada pelo indivíduo pode ter sua condição suprimida ou alterada com relação à presença dos limitantes.

Mutação Local: Na mutação local, o indivíduo sofre alteração nos valores que definem os limitantes das condições.

4.6.2.1 Mutação Estrutural

Neste tipo de mutação, o indivíduo sofre uma mudança *in-loco* e o *bit* de presença é ativado ou desativado, dependendo do estado inicial do *bit* de presença escolhido, *i.e.* caso *bit*=0, o limitante que está presente na condição passa a não ser mais considerado para compor o complexo da regra de conhecimento que aquele indivíduo representa. Dessa forma, um indivíduo pode ter o número de condições aumentada (a regra é especializada) ou diminuída (a regra é generalizada). Na Figura 4.8 é ilustrado esse processo. O indivíduo que representa a Regra 4.6 — Figura 4.8 **(a)** — tem um ponto escolhido aleatoriamente, o qual coincide com o *bit* de presença de uma condição daquele indivíduo e

o *bit* de presença é invertido — Figura 4.8 **(b)**. A Regra 4.7 é a nova regra de conhecimento criada por ocasião da mutação.

IF **salário** ≥ 2
AND **salário** $\leq 4{,}5$
AND **casado** = sim
THEN **empréstimo** = *sim*

Regra 4.6: Regra antes da mutação estrutural

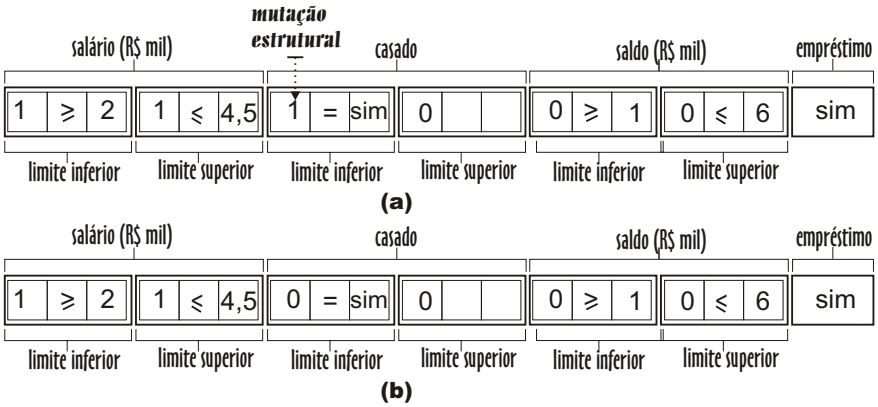

Figura 4.8: Operador de mutação em nível estrutural

IF **salário** ≥ 2
AND **salário** $\leq 4{,}5$
THEN **empréstimo** = *sim*

Regra 4.7: Regra após a mutação estrutural

É importante observar que caso a mutação torne presente um limitante que estava ausente, *i.e.* a condição passa a fazer parte do complexo da regras que aquele indivíduo representa, existe a possibilidade daquele limitante não ter sido sobrescrito pela condição correspondente da regra, o que pode acarretar a inclusão de uma condição inválida no complexo da regra. Dai a importância do processo de inicialização explicado na Seção 4.4.

4.6.2.2 Mutação Local

Neste tipo de mutação, o indivíduo sofre uma mudança *in-loco* e o resultado é uma alteração no valor de comparação da condição selecionada. Dessa forma, somente o valor de algum dos limitantes é alterado. A alteração é feita considerando se o atributo que participa da condição é qualitativo ou quantitativo. No caso de condições com atributos qualitativos, a mutação é feita de forma

que o valor passe a ser qualquer um dos outros possíveis valores que aquele atributo possa assumir. E, no caso de condições com atributos quantitativos, a mutação é feita de forma que o valor do atributo sofra uma alteração para mais ou para menos, dependendo de uma escolha aleatória. O valor a ser incrementado ou decrementado depende dos valores que esse atributo assume no conjunto de exemplos. Esse incremento (ou decremento) está dado pela menor diferença (em módulo) diferente de zero que existe entre dois valores quaisquer do atributo considerado.

Na Figura 4.9 é ilustrado esse processo. O indivíduo original dado pela Regra 4.8 — Figura 4.9 **(a)** — sofre mutação no limitante superior do atributo salário, passando a ser um indivíduo com outro valor no limitante superior do atributo salário — Figura 4.9 **(b)**. A Regra 4.9 é a regra de conhecimento criada por ocasião da mutação. Nesse exemplo, o valor utilizado para o decremento é a menor diferença encontrada para o atributo salário na Tabela 4.1 na página 49, $0,5$ neste caso.

IF **salário** $>$ 3
AND **salário** $\leq$ 4,5
AND **casado** = sim
THEN **empréstimo** = *sim*

Regra 4.8: Regra antes da mutação local

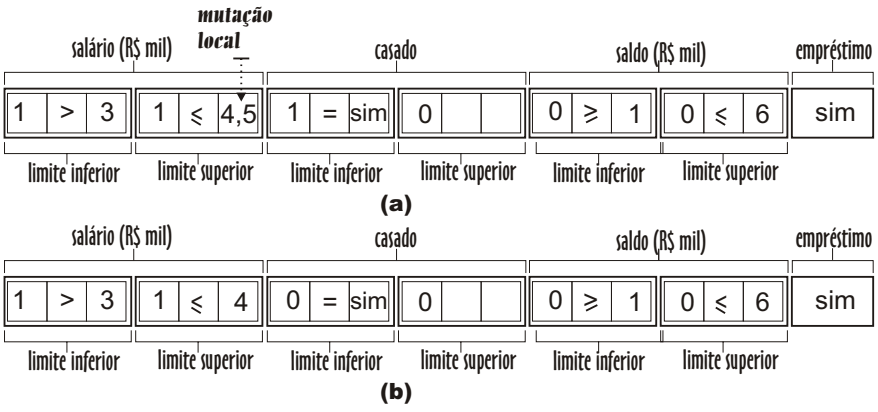

Figura 4.9: Operador de mutação em nível local

IF **salário** $>$ 3
AND **salário** $\leq$ 4
AND **casado** = sim
THEN **empréstimo** = *sim*

Regra 4.9: Regra após a mutação local

Pode-se observar que os diferentes tipos de mutação atuam em partes distintas da estrutura que define o indivíduo (regra). Dessa forma, consideramos interessante permitir que esses dois tipos de mutação atuem de forma concomitante em uma mesma população de regras.

Como novas regras são construídas por ocasião da ocorrência dos operadores evolutivos (*crossover* e mutação), os valores referentes às freqüências relativas das matrizes de contingência devem ser recalculadas antes da função de avaliação calcular os valores das medidas de avaliação de regras.

4.7 Função de Avaliação

Otimização se refere a encontrar a melhor solução possível para um problemas dado um conjunto de restrições. Quando o problema envolve a otimização de um único objetivo, deseja-se encontrar a melhor solução disponível, chamada de ótimo global, ou ao menos uma boa aproximação dessa solução. Contudo, são freqüentes os casos em que não existe somente um objetivo a ser otimizado, mas vários objetivos que na maioria das vezes são conflitantes (Coello et al., 2002). Esses problemas com vários objetivos são chamados multi-objetivo e requerem a utilização de métodos diferentes daqueles utilizados para solucionar problemas com um único objetivo. Na verdade, a noção de ótimo é diferente quando consideram-se problemas multi-objetivo porque neste caso é desejável um equilíbrio entre todos os objetivos ao invés de somente um objetivo atingir o ponto ótimo (Coello, 2006).

Em geral, os métodos para construir a função de avaliação de problemas multi-objetivo podem ser considerados em dois grandes grupos, aqueles que combinam os vários objetivos em uma função de avaliação simples-objetivo, e os que procuram pelo conjunto de soluções na fronteira de Pareto (Freitas, 2004), o qual consiste em encontrar o conjunto com o maior número de soluções não-dominadas e retornar esse conjunto de soluções não-dominadas para o usuário decidir qual a mais apropriada. O seguinte exemplo, freqüentemente citado na literatura, ilustra informalmente esse conceito — Figura 4.10.

O problema envolve a compra de um carro cujo custo seja mínimo e o conforto seja máximo. Existem 7 (sete) possíveis opções de compra, considerando esses dois objetivos. Analisando a Figura 4.10, pode-se descartar a opção **F**, pois a opção **B** oferece o mesmo conforto a um custo menor. Analogamente, pode-se descartar a opção **G**, pois a opção **E** oferece um conforto maior ao mesmo custo. Tem-se então 5 opções de compra: **A**, **B**, **C**, **D** e **E**. Em termos

quantitativos, essas 5 soluções concorrem entre si dado que nenhuma delas é melhor que a outra. Comparando aos pares, quando uma é melhor no custo o conforto não é bom e quando o conforto é alto, o custo não é adequado. Existe então um compromisso entre os objetivos, de forma que um dos objetivos não pode ser melhorado sem a deterioração do outro.

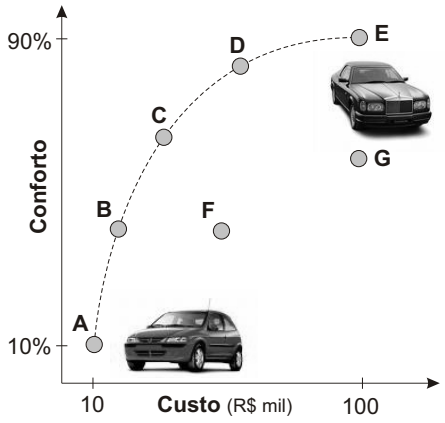

Figura 4.10: Exemplo da Fronteira de Pareto

Com base nesse compromisso existente entre os objetivos, pode-se dizer que uma solução domina a outra se os valores das funções objetivo que a primeira assume são todos melhores que o da segunda solução. Por exemplo, a solução **G** é dominada pelas soluções **C**, **D** e **E**. Também, a solução **F** é dominada pela solução **B**. As outras soluções que quando comparadas aos pares não se pode dizer qual é a melhor, dado os valores da função objetivo, são as chamadas soluções não dominadas. No exemplo, as soluções dominadas compõem o conjunto {F, G} e as soluções não dominadas compõem o conjunto {A, B, C, D, E}, chamadas de soluções Pareto-ótimas. As seguintes duas características descrevem esses dois conjuntos:

1. Quaisquer duas soluções pertencentes ao conjunto não dominado devem ser não dominadas uma em relação à outra.

2. Qualquer solução pertencente ao conjunto dominado deve ser dominada por pelo menos uma solução pertencente ao conjunto não dominado.

Os pontos contidos no conjunto de soluções não dominadas descrevem uma curva das soluções ótimas chamada de Fronteira de Pareto. Dessa forma, dado um problema multi-objetivo, ao obter a fronteira de Pareto, qualquer solução pertencente à fronteira satisfaz o problema inicial.

Nas soluções que combinam os vários objetivos em uma função de avaliação simples-objetivo, a idéia é atribuir pesos a cada objetivo e combinar esses critérios ponderados em uma função simples-objetivo. A combinação de critérios pode ser realizada utilizando as médias ponderadas (aritmética, Equação 4.3, ou harmônica, Equação 4.4, sendo T o número de termos envolvidos e t_i cada um dos temos) dos critérios considerados.

$$MA = \frac{\sum t_i}{T} \tag{4.3}$$

$$MH = \frac{T}{\sum \frac{1}{t_i}} \tag{4.4}$$

A média harmônica é interessante porque apresenta maiores valores quando os critérios envolvidos no cálculo apresentam um melhor equilíbrio. Na Figura 4.11 são mostradas as curvas resultantes do cálculo da média aritmética e da média harmônica sobre os pares de valores $(x, y) \,|\, x \in \{1..10\},\, y \in \{7\}$. O número 7 foi arbitrariamente escolhido como parâmetro de controle na comparação. No eixo das abscissas está representada a variação do valor de x utilizado no cálculo da média conjuntamente com o valor 7. No eixo das ordenadas está representado os valores obtidos das médias aritmética e harmônica para cada par de valores considerado. Pode ser observado que a média harmônica assume valores próximos ao da média aritmética quando os valores considerados, para esse caso, estão próximos ao valor 7. Valores muito discrepantes resultam numa média harmônica inferior à média aritmética. Por exemplo, o par $(1, 7)$ produz valor 4 para a média aritmética, enquanto que para a média harmônica produz um valor em torno de 1,8.

Neste trabalho adotamos soluções que combinam os vários objetivos, utilizando tanto a média aritmética quanto a média harmônica, em uma função de avaliação simples-objetivo. Duas soluções são consideradas. Na primeira, as medidas de avaliação de regras podem ser combinadas, atribuindo pesos (ou não) a cada uma delas. Esta é uma solução muito simples na qual a interação entre os vários objetivos não é necessariamente levada em conta pela função multi-objetivo. Um outro problema desta solução, é que elas devem considerar os mesmos tipo de medidas de avaliação. O problema não consiste somente em normalizar os valores das medidas individuais, se for o caso, mas também que elas sejam compatíveis. Por exemplo, considerando a medida de Laplace conjuntamente com a medida de complexidade sintática de uma regra, produz um valor de função de avaliação que não tem o menor significado para o usuário. Este problema pode ser solucionado utilizando *rankings*, que é a segunda solução proposta.

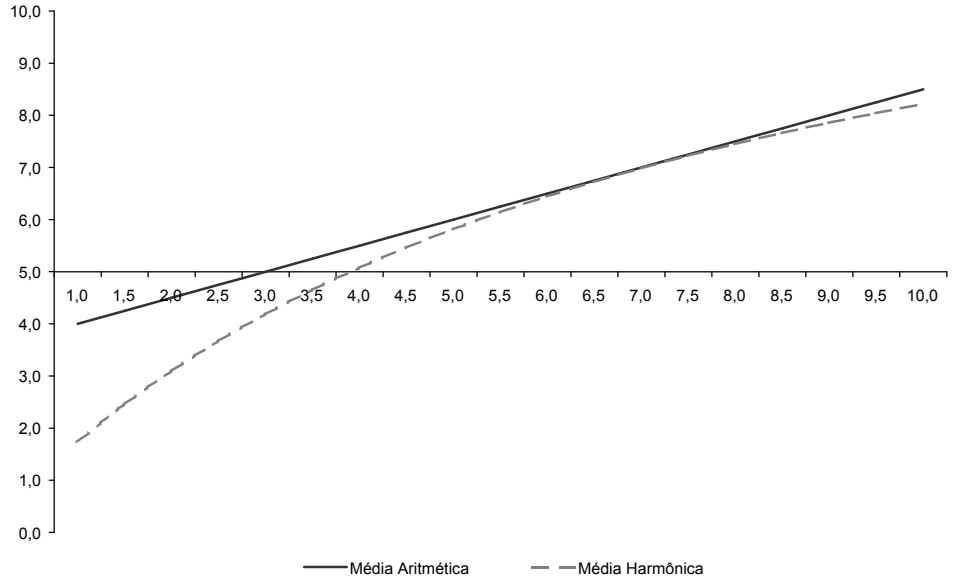

Figura 4.11: Média aritmética *vs.* média harmônica

Quando um número de indivíduos esta organizado em ordem de alguma medida, isso configura um *ranking* (Kendall, 1990). É importante ressaltar que o *rank* do indivíduo não considera a magnitude da medida que deu origem ao *rank* e nem a escala da medida. Os indivíduos são arranjados de forma que é considerada apenas a ordem do valor da medida original. Assim, sabe-se que o indivíduo com *rank* 1 é melhor que o indivíduo com *rank* 2. Em caso de empate, ou seja, indivíduos com o mesmo valor de medida, os *ranks* iniciais são substituídos pela média aritmética dos valores iniciais dos ranks. Por exemplo, se os indivíduos que deveriam ter *rank* 3 e 4 possuírem a mesma medida (empate), o *rank* de ambos será $\frac{1}{2}(3+4) = 3,5$. O critério de desempate é importante por manter a distribuição dos *ranks* uniforme em toda a população de indivíduos e a soma dos *ranks* como uma constante.

Uma função multi-objetivo pode ser construída levando em consideração essa idéia. As medidas de avaliação de regras são utilizadas individualmente para construir um *rank*. Então, com os valores dos *ranks* de cada indivíduo é calculada uma média (ponderada ou harmônica com pesos) que origina uma nova medida. Essa nova medida é utilizada como função de avaliação simples-objetivo.

Por exemplo, considere os seguintes dez indivíduos e seus *ranks*, igualmente ponderados, com relação a três medidas A, B e C quaisquer, mostrados na Tabela 4.2.

Nesse exemplo, a função de avaliação simples-objetivo construída conside-

Indivíduos	A	*Rank*(A)	B	*Rank*(B)	C	*Rank*(C)	Função de Avaliação
I_1	0.9	2	130	1	0.10	7	3.3
I_2	0.6	5	20	9	0.01	10	8.0
I_3	0.8	3	100	2.5	0.25	1.5	2.3
I_4	0.2	10	5	10	0.20	5	8.3
I_5	0.5	7	50	6.5	0.09	8	7.2
I_6	0.4	9	30	8	0.15	6	7.6
I_7	0.5	7	60	5	0.22	4	5.3
I_8	1.0	1	100	2.5	0.25	1.5	1.6
I_9	0.5	7	50	6.5	0.05	9	7.5
I_{10}	0.7	4	70	4	0.23	3	3.6

Tabela 4.2: Exemplo de combinação de *ranks*

rando os *ranks* individuais das três medidas A, B e C, indica que I_8 é o melhor indivíduo, I_3 é o segundo melhor indivíduo e assim por diante. Como pode ser observado, a função de avaliação é tal que os valores das medidas individuais encontram-se melhor distribuídas nos indivíduos considerados melhores, que é um dos objetivos deste trabalho.

Ainda que não seja possível afirmar que o melhor indivíduo escolhido utilizando este método seja um indivíduo que pertence à fronteira de Pareto, o uso de *rankings* vem recebendo uma grande atenção da comunidade de Inteligência Artificial nos últimos anos (Agarwal et al., 2005). Um aspecto importante tem a ver com a possibilidade de utilizar uma combinação de medidas (ou propriedades) que podem ser de qualquer tipo. Também, como as medidas individuais são usadas somente para *rankear* os indivíduos, não há problema de escala entre elas. Essas e outras facilidades, tais como a simplicidade de implementação, têm impulsionado a pesquisa sobre *rankings* pela comunidade (Prati, 2006; Huang & Ling, 2006).

Neste trabalho propomos a combinação ponderada dos *rankings* das medidas individuais de regras usando a média aritmética ou a média harmônica, em uma função simples-objetivo.

4.8 Convergência do Algoritmo

Quando é fornecida ao algoritmo evolutivo uma população inicial de indivíduos, neste trabalho composta por regras de conhecimento, uma nova população é criada pela atuação do método de seleção e dos operadores de *crossover* e mutação, e essa nova população deve ser avaliada para verificar a aptidão dos indivíduos. Em geral, na população obtida tem-se um indivíduo com valor da função de avaliação melhor que a do melhor indivíduo da população inicial. Entretanto, para verificar se essa solução é a melhor possível de ser obtida, o

ciclo evolutivo deve ser executado várias vezes até que haja uma estabilidade da melhor solução encontrada, *i.e.* o melhor indivíduo é semelhante entre várias gerações.

Neste trabalho propomos verificar a convergência utilizando um intervalo de 20 (vinte) gerações como referência. Assim, o ciclo evolutivo deve ser executado 20 vezes, sendo que a partir da 21^a execução o algoritmo evolutivo começa a verificar se ocorreu a convergência (últimas 20 melhores soluções, uma de cada ciclo, não sofreu alteração no valor da aptidão). Propomos realizar essa verificação calculando o desvio padrão dos valores da função de avaliação do melhor indivíduo dessas últimas 20 soluções. Se o valor do desvio padrão for menor que um limiar estabelecido, entende-se que as soluções obtidas nas últimas 20 gerações estão muito próximas (ou são as mesmas) e o algoritmo evolutivo não consegue evoluir outra solução que seja significativamente melhor que as anteriores. Caso contrário, uma nova geração é criada e a convergência do algoritmo é verificada incluindo agora esse melhor indivíduo no intervalo das últimas 20 soluções.

Neste trabalho foi escolhido 0.005 como valor padrão (*default*) do limiar. A nossa proposta foi implementada em um sistema computacional — $\mathcal{ECLE}$ — descrito no próximo capítulo. Utilizando a $\mathcal{ECLE}$, foram realizados diversos experimentos com o algoritmo evolutivo utilizando vários conjuntos de dados diferentes com o objetivo de encontrar um valor de limiar padrão, bem como o tamanho do intervalo mais apropriados, para o cálculo do desvio padrão das últimas soluções. Esses experimentos apontaram um intervalo de tamanho 20 com limiar padrão de 0.005[3].

Para ilustrar a convergência do algoritmo evolutivo utilizando o critério padrão (intervalo=20 e limiar=0.005), na Figura 4.12 é apresentado o gráfico de convergência utilizando a medida de suporte (simples-objetivo) para o conjunto de dados breast, o qual foi construído utilizando as facilidades implementadas na $\mathcal{ECLE}$. Nesse gráfico, no eixo das abscissas está o número de gerações enquanto que no eixo das ordenadas está o valor da medida considerada. São apresentadas nesse gráfico duas curvas, uma para o valor médio da medida na população e outra que é o valor da medida do melhor indivíduo. Nesse exemplo, a partir da 7^a geração a melhor solução não muda, ocasionando a convergência do algoritmo em torno da 21^a geração.

Já para o gráfico de convergência ilustrado na Figura 4.13, foram utilizadas as medidas de Laplace e suporte para compor uma função multi-objetivo utilizando *rankings*. Como no exemplo anterior, para cada medida são apresentadas

[3]Esses são os valores padrão (*default*) usados pela $\mathcal{ECLE}$, mas também podem ser especificados pelo usuário.

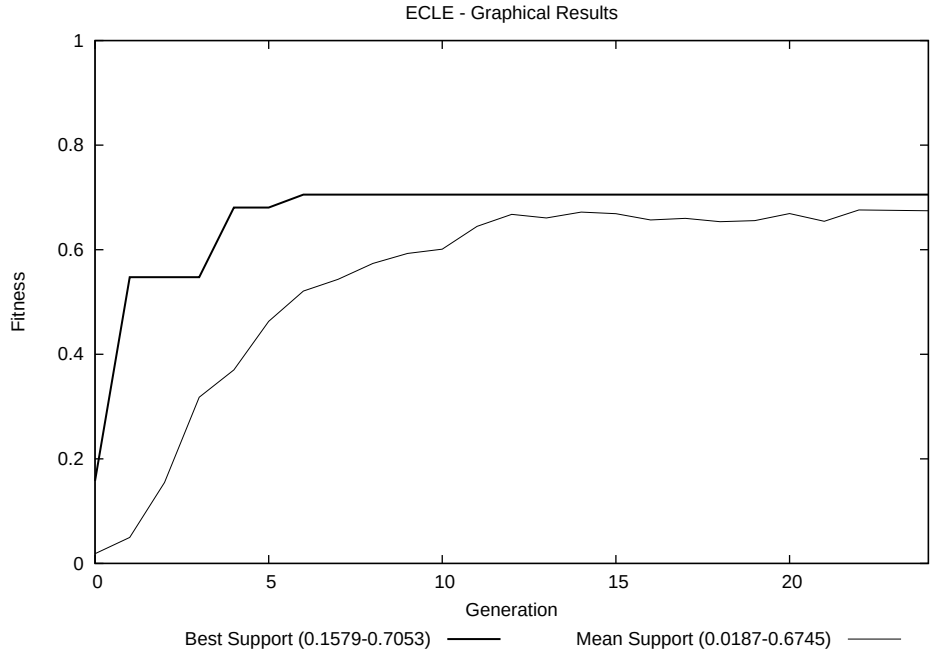

Figura 4.12: Exemplo de convergência do algoritmo evolutivo proposto – breast – simples-objetivo

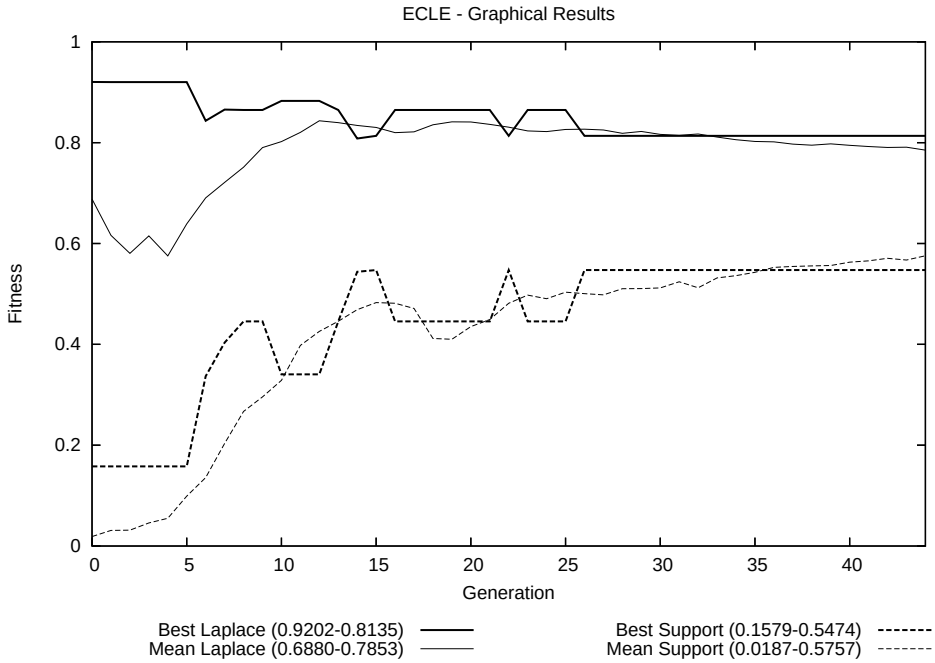

Figura 4.13: Exemplo de convergência do algoritmo evolutivo proposto – breast – multi-objetivo

duas curvas, uma do valor médio da medida da população e outra do valor dessa medida no melhor indivíduo. É possível verificar as tentativas para esta-

belecer o equilíbrio entre as medidas que participam da função multi-objetivo (não representada nesse gráfico). Ambas medidas estabilizam na 26^a iteração, ocasionando a convergência na 43^a iteração.

Como mencionado, quando o algoritmo evolutivo converge, pode ser que na população final exista mais que um indivíduo que possua o mesmo valor da função de avaliação. Nesse caso, se o usuário solicitar uma única regra, é utilizado um critério de desempate para decidir qual dos indivíduos empatados é apresentado como solução. Neste trabalho, assim como no método de seleção por elitismo, o desempate é feito por uma escolha aleatória entre os indivíduos empatados. Caso o usuário não solicite uma única regra, todas as regras empatadas com o melhor valor da função de avaliação são mostradas.

4.9 Considerações Finais

Neste capítulo foram apresentados a estrutura de dados que comporta a representação dos indivíduos enquanto regra de conhecimento, como os operadores evolutivos podem trabalhar sobre essa estrutura e como a aptidão desses indivíduos pode ser calculada utilizando a função de avaliação em termos das medidas de qualidade de regras. Como pode ser observado, a nossa proposta apresenta um campo de trabalho vasto para novas definições e reformulações dos operadores evolutivos e da função de avaliação. Também, foi apresentado o critério de convergência utilizado para verificar a estabilidade da solução.

Capítulo

5

Projeto da Biblioteca Evolutionary Computing Learning Environment

Quem estuda e não pratica o que aprendeu é como o homem que lavra e não semeia.
Provérbio Árabe

5.1 Considerações Iniciais

Com o objetivo de avaliar a nossa proposta do algoritmo evolutivo para construir regras de conhecimento com propriedades específicas, foi projetada e implementada a biblioteca Evolutionary Computing Learning Environment — $\mathcal{ECLE}$. Essa biblioteca faz uso de diversas facilidades existentes no ambiente DISCOVER, que é um projeto de grande porte em desenvolvimento no nosso laboratório de pesquisa — LABIC — voltado para aquisição automática de conhecimento. A $\mathcal{ECLE}$ encontra-se atualmente

inserida nesse ambiente computacional e foi projetada e implementada seguindo o paradigma de orientação à objetos. Além de implementar o algoritmo evolutivo proposto, permitindo diferentes métodos de seleção, bem como uma vasta gama de parâmetros de aplicação dos operadores evolutivos e funções de avaliação, a $\mathcal{ECLE}$ também possui facilidades para realizar avaliação experimental dos resultados obtidos.

5.2 Arquitetura Geral

Qualquer regra de conhecimento simbólica do tipo *if* <*complexo*> *then* <*classe*>, relacionada a um conjunto de exemplos de um mesmo domínio $\mathcal{D}$, pode ser transformada para a estrutura de indivíduos utilizada para construir a população inicial do algoritmo evolutivo. Como mencionado, essas regras podem ser regras induzidas por diversos classificadores simbólicos, regras criadas por outro tipo de algoritmo e/ou regras fornecidas pelo próprio usuário. Utilizando alguma das facilidades existentes no ambiente DISCOVER, descritas brevemente na Seção 5.3, essas regras podem ser transformadas e armazenadas em uma base inicial regras na sintaxe $\mathcal{PBM}$, descrita no capítulo anterior, como ilustrado na Figura 5.1.

Para isso, deve ser utilizado um conjunto de exemplos relacionados ao mesmo domínio $\mathcal{D}$ das regras iniciais, mas ele não necessita ser idêntico ao(s) utilizado(s) para gerar as regras de conhecimento iniciais. Esse conjunto de exemplos deve estar expresso na sintaxe padrão utilizada no ambiente DISCOVER, sintaxe DSX (Batista, 2001), que possuí facilidades para, dado um conjunto de exemplos na sintaxe DSX, realizar a padronização e verificar a cobertura das regras iniciais. Em outras palavras, utilizando como entrada os conjuntos de regras iniciais, elas podem ser transformadas para o formato $\mathcal{PBM}$, utilizando um conjunto de exemplos do mesmo domínio $\mathcal{D}$, na sintaxe DSX, para calcular a matriz de contingência de cada regra. Assim, utilizando as facilidades existentes no DISCOVER, os conjuntos de regras iniciais fornecidos pelo usuário são transformados para a sintaxe $\mathcal{PBM}$ e armazenados na base de regras a ser considerada pela $\mathcal{ECLE}$. A arquitetura geral para aplicação da $\mathcal{ECLE}$ está ilustrada na Figura 5.2.

As regras no formato $\mathcal{PBM}$ são transformadas pela $\mathcal{ECLE}$ para a estrutura de indivíduos do algoritmo evolutivo definido na Seção 4.4 na página 46. Deve ser observado que os indivíduos iniciais podem ser constituídos pela totalidade das regras na base de regras iniciais fornecidas pelo usuário, ou por um subconjunto (amostra) dessas regras. Após construir a população inicial, o

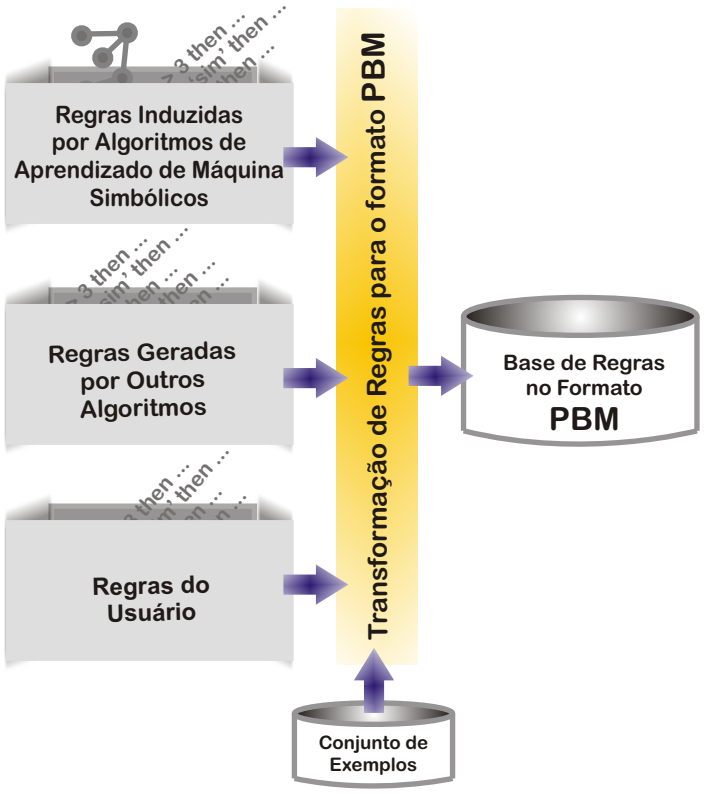

Figura 5.1: Construção da base inicial de regras no formato $\mathcal{PBM}$

algoritmo evolutivo é ativado para construir regras de conhecimento com propriedades especificadas pelo usuário. Durante a sua execução, para avaliar as regras o algoritmo evolutivo implementado faz uso de um conjunto de exemplos fornecido pelo usuário, também expresso na sintaxe DSX do DISCOVER, que não necessita ser idêntico aos conjuntos utilizados para criar as regras iniciais, ou ao conjunto de exemplos utilizado para transformar essas regras iniciais para o formato $\mathcal{PBM}$ — Figura 5.1. A única restrição é que o domínio $\mathcal{D}$ desse conjunto de exemplos seja o mesmo dos anteriores.

5.3 O Projeto DISCOVER

Nos últimos anos, diversos trabalhos na área de aquisição automática de conhecimento, a maioria dos quais são realizados de forma cooperativa, vêm sendo desenvolvidos no nosso laboratório de pesquisa LABIC. Com o decorrer dos anos, foram observadas algumas sobreposições de implementações. Esses, e outros fatores, motivaram alguns pesquisadores a propor a implementação de um ambiente computacional denominado DISCOVER (Baranauskas &

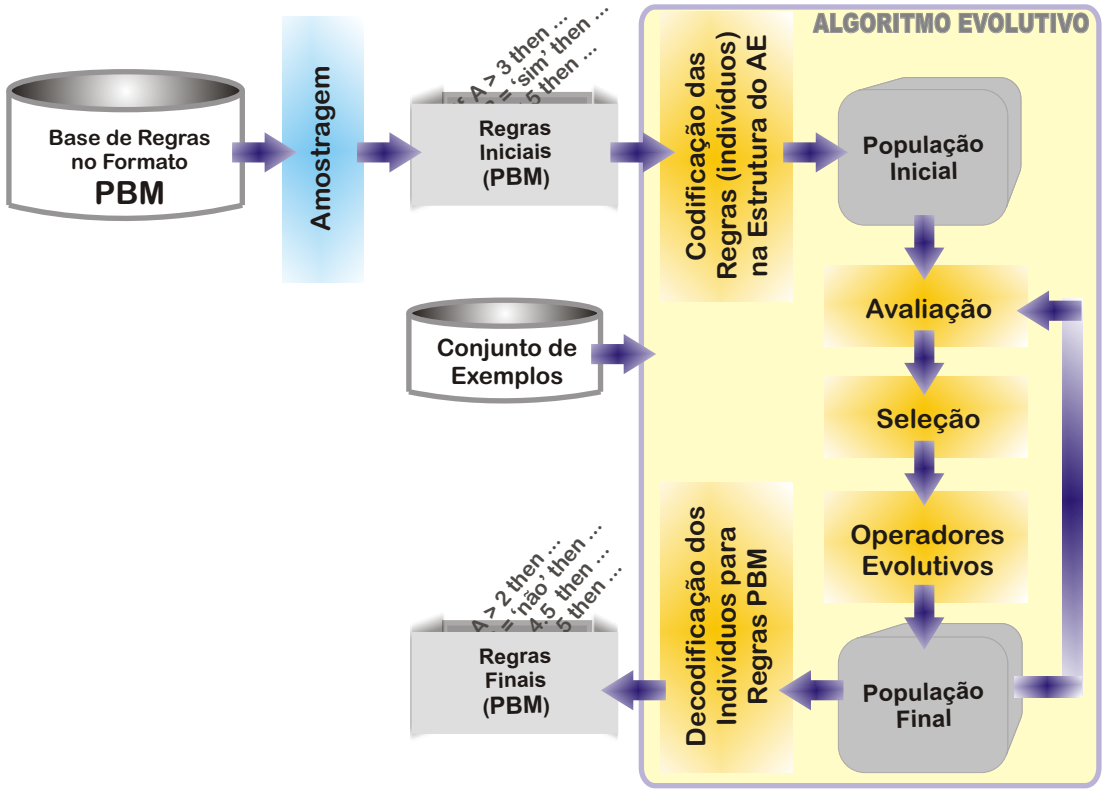

Figura 5.2: Funcionamento da $\mathcal{ECLE}$

Batista, 2000).

O objetivo principal do projeto DISCOVER é reduzir problemas de implementação e integração de ferramentas e algoritmos para apoiar todas as etapas do processo de descoberta de conhecimento. O DISCOVER pode ser entendido como um conjunto de métodos para manipular dados e conhecimento por meio de sintaxes padrão (Batista & Monard, 2003b; Batista, 2001; Prati et al., 2001) para a representação de dados e do conhecimento induzido por diferentes algoritmos de aprendizado, bem como por bibliotecas que oferecem um conjunto de funcionalidades básicas para tais manipulações. Utilizando o paradigma orientado a objetos na sua implementação, as bibliotecas de classes desenvolvidas em PERL (Wall et al., 1996), podem ser integradas pelos usuários do ambiente. Um *framework* para a integração dos componentes do ambiente DISCOVER foi proposta por Prati (2003) utilizando *software patterns*, no qual os componentes são integrados por meio de uma linguagem baseada em XML, a qual foi denominada xDML.

O DISCOVER conta com vários ambientes, tais como o ambiente Discover Learning Environment — DLE — composto pela biblioteca de classes Discover Object Library — DOL — e o ambiente para gerenciamento de experimentos SNIFFER,

que foram desenvolvidos para as diferentes tarefas de pré-processamento de dados no formato atributo-valor e análise dos classificadores induzidos por algoritmos de aprendizado supervisionado utilizados no processo de aquisição de conhecimento (Batista, 2003). Novas funcionalidades estão sendo especificadas para regressão e regras de associação, bem como para algoritmos de aprendizado não-supervisionado (Metz, 2006).

Na Figura 5.3 é mostrado, de uma forma simplificada, como os filtros, sintaxes e bibliotecas interagem uns com os outros para o caso de algoritmos simbólicos proposicionais de aprendizado supervisionado. Os dados, no formato atributo-valor na sintaxe padrão DSX do DISCOVER, podem ser submetidos a diversos processos de pré-processamento já implementados (Batista, 2003; Batista & Monard, 2003a; Batista et al., 2004b; Prati et al., 2004; Voltolini, 2006). Feito isso, esses dados são convertidos para a sintaxe padrão do algoritmo de aprendizado que o usuário deseja executar. O conjunto de regras de conhecimento (classificador simbólico) induzido por esse algoritmo de aprendizado, é posteriormente convertido para a sintaxe padrão $\mathcal{PBM}$ de representação de conceitos proposicionais (regras) do DISCOVER (Prati et al., 2001). Esse conhecimento pode ser posteriormente submetido a diversos processos de pós-processamento de conhecimento implementados e que estão sendo implementados. As linhas tracejadas na Figura 5.3 mostram os processos que utilizam a biblioteca de classes DOL e rotinas implementadas no ambiente DLE do DISCOVER que contém métodos para transformar regras de conhecimento para a sintaxe padrão do DISCOVER — $\mathcal{PBM}$ — bem como para avaliar essas regras utilizando um conjunto de exemplos. Essas funcionalidades foram utilizadas no desenvolvimento deste trabalho e são brevemente descritas a seguir.

5.3.1 A Biblioteca de Classes Discover Object Library — DOL

A biblioteca DOL é uma biblioteca orientada a objeto baseada em padrões de projeto (Gamma et al., 1995). As classes dessa biblioteca implementam as tarefas de manipulação e gerenciamento de dados mais comuns em pré-processamento, tais como gerenciamento de diferentes sintaxes de arquivos de dados e atributos, amostragens, métodos de *resampling*, estatísticas descritivas, normalizações de dados, entre outros (Batista & Monard, 2003b). O objetivo da DOL é dar suporte à criação de novos métodos de pré-processamento de dados e ser uma biblioteca de métodos de pré-processamento.

O conjunto de métodos da DOL permite ao desenvolvedor uma forma simples de ter acesso aos dados, os quais podem estar disponíveis em arquivos

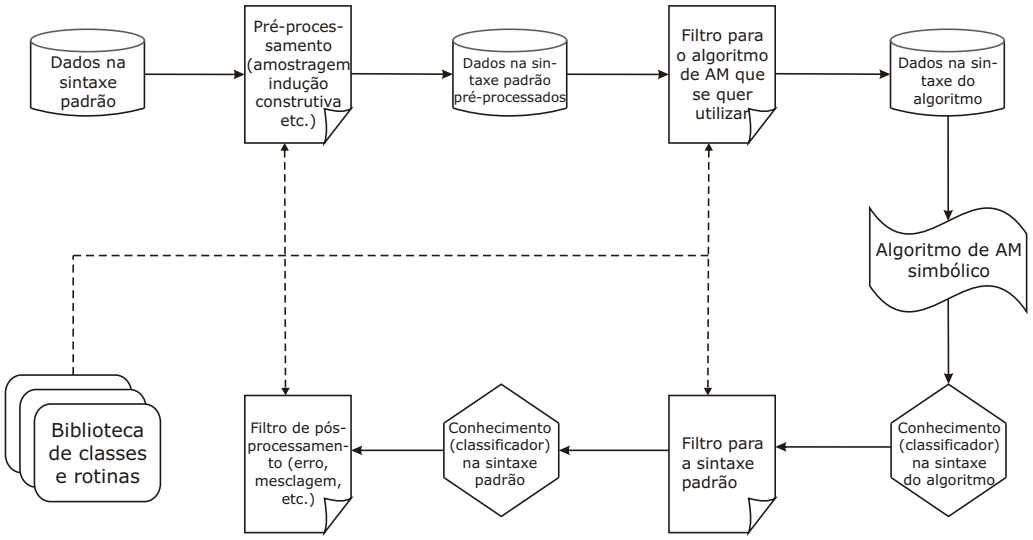

Figura 5.3: Interação entre filtros, sintaxes e bibliotecas do DISCOVER (Chiara, 2003)

texto ou em tabelas de banco de dados relacionais. Os dados são carregados em uma estrutura que pode ser acessada por uma variedade de métodos implementados para manipular essa estrutura. Após, ela pode ser novamente armazenada em arquivos texto, em diferentes sintaxes, ou carregados em uma tabela de banco de dados relacional. Entre as principais funcionalidades da biblioteca DOL pode-se citar: manipulação de atributos e dados, integração com diversos sistemas de aprendizado de máquina, integração com sistemas gerenciadores de banco de dados, filtros de exemplos, estatísticas descritivas e correlações, além de diversos métodos de *resampling*. A biblioteca DOL foi desenvolvida em uma arquitetura modular na qual cada módulo é constituído de uma ou mais classes que realizam um conjunto bem definido de tarefas. A biblioteca possui um módulo central chamado **Core**, o qual carrega um conjunto de dados em uma estrutura e disponibiliza, atualmente, mais de 60 métodos capazes de consultar e manipular essa estrutura. O módulo **Core** é o único que precisa obrigatoriamente ser carregado por uma aplicação que utiliza a biblioteca, os demais módulos são carregados apenas quando for necessário.

5.3.2 O Ambiente Computacional SNIFFER

O ambiente computacional SNIFFER é um ambiente de gerenciamento de avaliações e comparações experimentais de algoritmos de AM. O ambiente computacional SNIFFER automatiza a avaliação experimental, e está atualmente

integrado com diversos sistemas de AM simbólico, entre eles o $\mathcal{C}4.5$ (Quinlan, 1988) e o $\mathcal{CN}2$ (Clark & Boswell, 1991) utilizados neste trabalho.

O ambiente SNIFFER complementa a biblioteca DOL pois permite que diferentes indutores proposicionais, bem como diferentes métodos de pré-processamento de dados, sejam avaliados e comparados experimentalmente de uma forma rápida e segura (Batista & Monard, 2003b). As comparações são realizadas utilizando testes estatísticos de significância, permitindo identificar quando um método supera outro com 95% ou 99% de confiança. Deve ser observado que o SNIFFER provê uma *API* (*Application Programming Interface*) que permite estender as funcionalidades do ambiente. Por exemplo, essa *API* pode ser utilizada para avaliar sistemas de aprendizado utilizando outras medidas de desempenho ou implementar novos testes estatísticos de significância. O ambiente SNIFFERECLE, implementado neste trabalho, e descrito brevemente na Seção 5.8, para realizar avaliações do algoritmo evolutivo proposto, utiliza das funcionalidades deste ambiente.

5.4 A Biblioteca de Classes Evolutionary Computing Learning Environment — $\mathcal{ECLE}$

A biblioteca $\mathcal{ECLE}$, implementada neste trabalho, tal qual a DOL, é uma biblioteca de classes orientadas a objeto. As classes dessa biblioteca implementam os diversos métodos que compõem o algoritmo evolutivo por nós proposto para a construção de regras com propriedades específicas, tais como métodos de seleção, *crossover*, mutação e avaliação. O objetivo da $\mathcal{ECLE}$ é fornecer suporte à execução de experimentos com diversos tipos de parâmetros e métodos existentes na área de computação evolutiva aplicada ao aprendizado de regras de conhecimento com propriedades específicas.

Por ser um conjunto de classes orientadas a objeto, a $\mathcal{ECLE}$ permite ao desenvolvedor sua fácil extensão e a implementação de novos métodos que são automaticamente reconhecidos pelas demais classes que compõem essa biblioteca. Toda a biblioteca foi projetada e implementada segundo o paradigma de orientação a objetos. Cada funcionalidade da $\mathcal{ECLE}$ é empacotada numa classe bem definida, conforme diagrama UML ilustrado na Figura 5.4 (Booch et al., 1998). Todas as classes da $\mathcal{ECLE}$ podem ser agrupadas em oito conjuntos distintos, de acordo com as funcionalidades implementadas em cada uma delas. São conjuntos da $\mathcal{ECLE}$:

1. indivíduos e população;

2. funções de avaliação;
3. métodos de seleção;
4. operadores de *crossover*;
5. operadores de mutação;
6. estatísticas;
7. iteradores;
8. módulo de automatização.

A seguir, são brevemente descritas as classes e métodos mais importantes relacionados à esses conjuntos, cujo diagrama de classes é ilustrado na Figura 5.4. Informações detalhadas sobre a modelagem das classes, seus atributos e métodos que cada uma delas provê, podem ser consultadas em Giusti et al. (2006).

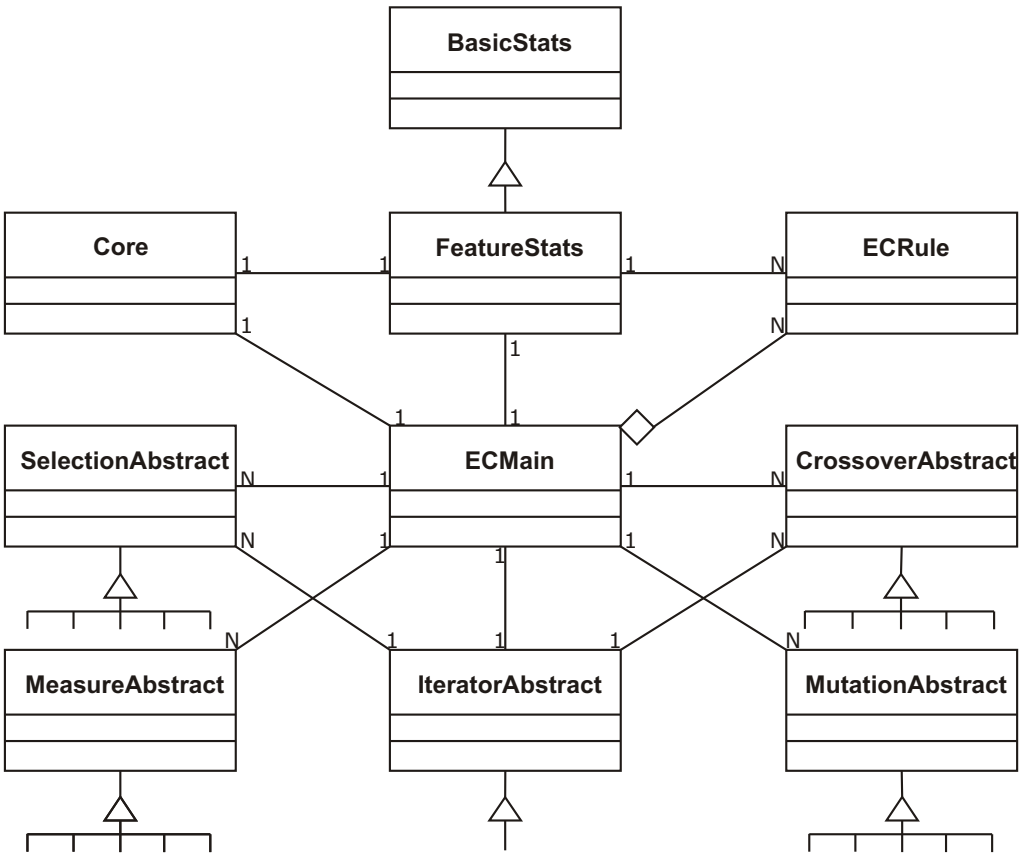

Figura 5.4: Diagrama de classes da $\mathcal{ECLE}$. As classes abstratas possuem classes descendentes que foram omitidas para fins de simplificação.

Core: é uma classe da DOL responsável por armazenar todos os exemplos presentes no conjunto de dados fornecido como entrada. Uma vez que os exemplos estejam em um objeto **Core**, nenhuma referência é feita ao arquivo físico, mas sempre como objeto e os métodos que a classe **Core** provê.

A classe ECRule: armazena e manipula uma quantidade bastante variada de informações de cada regra (indivíduo). Além do conteúdo genético ($< complexo >$), esta classe armazena também a matriz de contingência, a cabeça da regra (classe predita) e o ID da regra (número inteiro). Está classe também fornece os métodos de *interface* entre as demais classes da $\mathcal{ECLE}$ e outras fornecidas pelo DISCOVER.

A classe ECMain: armazena e permite acesso fácil às populações de indivíduos utilizadas pelo algoritmo evolutivo. A classe armazena três populações, a saber: população inicial, população intermediária e população final.

A classe SelectionAbstract: implementa uma série de métodos que são herdados e sobrescritos utilizando polimorfismo. Todos os métodos de seleção: elitismo, roda da roleta, torneio, *ranking* linear e exponencial são implementados em classes descendentes da SelectionAbstract.

A classe CrossoverAbstract: implementa os dois principais métodos para as classes do conjunto de operadores de *crossover*. Os operadores de *crossover* do algoritmo evolutivo proposto neste trabalho: estrutural, nível de atributo e local, são implementados em classes descendentes da classe CrossoverAbstract. No caso do *crossover*, existe uma classe descendente chamada CompositeCrossover que é responsável por gerenciar a aplicação de qualquer combinação dos tipos de *crossover* existentes. Na implementação feita o usuário pode escolher se o *crossover* será realizado por pais que predizem qualquer classe ou apenas por pais que predizem a mesma classe. O padrão da $\mathcal{ECLE}$ é aplicar *crossover* apenas a pais que predizem a mesma classe.

A classe MutationAbstract: implementa os dois principais métodos para as classes do conjunto de operadores de mutação. Tal como na classe abstrata para os operadores de *crossover*, os operadores de mutação são implementados em classes descendentes da classe MutationAbstract. Também, como no *crossover*, existe uma classe descendente CompositeMutation que é responsável por gerenciar a aplicação de qualquer combinação dos tipos de mutação existentes.

A classe MeasureAbstract: é utilizada para executar o cálculo das medidas de todos os indivíduos da população. Além do cálculo das medidas (aptidão dos indivíduos), os objetos dessa classe armazenam na forma de atributos os valores de aptidão ótimo, médio e pior. Todas as medidas de avaliação de regras são implementadas como classes descendentes da MeasureAbstract. No caso da função de avaliação ser multi-objetivo, existem duas classes responsáveis pela combinação das medidas utilizando média ponderada ou utilizando *rankings*, as classes CompositeMeasure e CompositeMORankingMeasure, respectivamente. A classe MeasureAbstract ainda utiliza duas classes que são responsáveis pelo cálculo da composição das médias ponderadas, podendo ser feita utilizando média aritmética ou média harmônica, com as classes WeightedMean e HarmonicMean, respectivamente.

A Classe FeatureStats: é uma classe descendente da classe **BasicStats** por herança. A classe **BasicStats** faz parte da DOL e também fornece estatísticas sobre o conjunto de dados. A diferença é que a primeira tem um foco muito mais voltado aos atributos do que aos valores da amostra. A classe **FeatureStats** abstrai uma série de passos para a obtenção de valores e os oferece para todo o sistema através de métodos de acesso, os quais fornecem, entre outros, médias (para atributos quantitativos), modas (para atributos qualitativos) e valores aleatórios em função da distribuição desses valores entre as classes do conjunto de dados.

A classe IteratorAbstract: possui a maioria dos métodos para que o algoritmo evolutivo funcione a contento. É nesta classe que ocorre a ordenação lógica da chamada de todos os demais métodos apresentados: seleção, *crossover*, mutação, medidas e estatísticas. Os critérios de parada também são definidos nesta classe. Existem duas classes descendente diretas desta classe, as quais são responsáveis por executar o algoritmo evolutivo para obter uma única regra ou para obter um classificador, RegularIterator e ClassifierIterator, respectivamente.

5.5 Esquema de Funcionamento da $\mathcal{ECLE}$

Com o objetivo de ilustrar o funcionamento da $\mathcal{ECLE}$, na Figura 5.5 é mostrada quais são as entradas necessárias para a execução do algoritmo evolutivo de um experimento e qual a saída que será obtida. Como entrada, quatro arquivos são essenciais para a $\mathcal{ECLE}$, os quais, por *default*, possuem o mesmo

nome e são reconhecidos pela sua extensão (.data, .names, .rules e .conf): um arquivo de dados (<arq>.data), um arquivo de nomes (<arq>.names)[1], um ou mais arquivos de regras (<arq>.rules) e um arquivo de configuração (<arq>.conf), descritos a seguir.

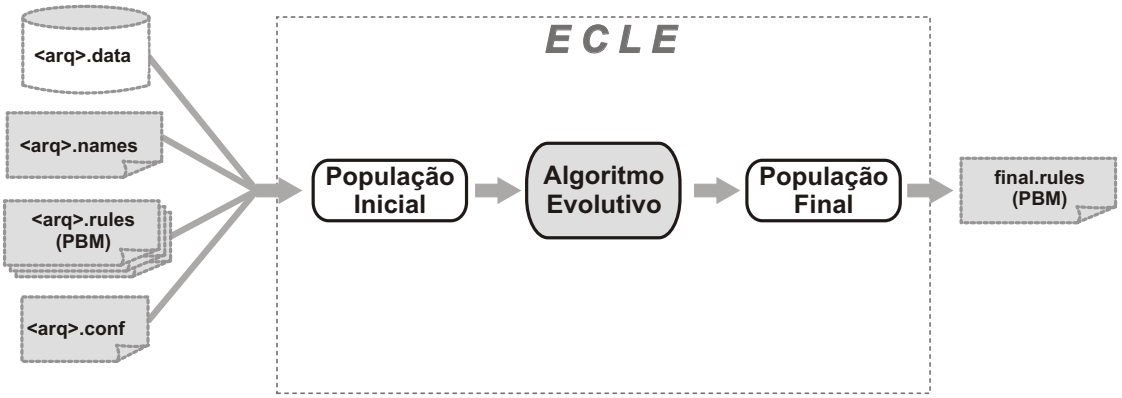

Figura 5.5: Esquema de funcionamento da $\mathcal{ECLE}$

Arquivo de dados (<arq>.data): conjunto de exemplos a ser analisado.

Arquivo de nomes (<arq>.names): utilizado para definir e analisar os atributos do conjunto de exemplos.

Arquivo de regras (<arq>.rules): pelo menos um arquivo de regras deve ser fornecido, o qual por *default* possui o mesmo nome do arquivo de dados. Podem ser fornecidos outros arquivos de regras, com outros nomes, desde que sejam especificados no arquivo de configuração. As regras devem estar no formato $\mathcal{PBM}$.

Arquivo de configuração (<arq>.conf): contém as instruções para que a $\mathcal{ECLE}$ possa selecionar as populações de indivíduos do experimento e quais operadores evolutivos aplicar, bem como os métodos de avaliação e os critérios de parada.

Como saída a $\mathcal{ECLE}$ fornece um arquivo (o nome padrão do arquivo é **final.rules**) que contém a população final de regras já no formato $\mathcal{PBM}$. Vale destacar que ao final do processo descrito, a $\mathcal{ECLE}$ gera um arquivo no formato *PostScript* contendo o gráfico da convergência das diferentes medidas consideradas, quer seja simples-objetivo ou multi-objetivo, como os utilizados na Seção 4.8 na página 63.

[1]Como mencionado, o conjunto de dados deve estar no formato DSX do DISCOVER, o qual requer ambos arquivos .names e .data conjuntamente(Batista, 2001).

5.6 Configuração e Comandos da $\mathcal{ECLE}$

É importante destacar que a execução do algoritmo evolutivo é definida pela informação no arquivo de configuração. Cada linha do arquivo de configuração refere-se a um comando para, entre outros, configurar qualquer dos parâmetros do algoritmo evolutivo, tais como o método de seleção a ser utilizado, os operadores de *crossover* e mutação, a função de avaliação (simples ou multi-objetivo) e os critérios de parada (máximo de iterações, convergência, ótimo alcançado). A construção do arquivo de configuração é bastante intuitiva. A linguagem do arquivo de configuração possui uma gramática por nós definida e implementada, a qual é reconhecida por uma das classes que faz parte da $\mathcal{ECLE}$. Segue uma breve descrição do formato dos comandos e funcionalidades suportados pela $\mathcal{ECLE}$.

verbose: A $\mathcal{ECLE}$ é capaz de gerar informação de depuração para todas as operações que executa. O comando verbose faz com que os comandos tenham ou não sua execução registrada. Opcionalmente, pode-se informar o nome do arquivo de saída (o arquivo padrão é **experimento.verbose**).

verbose on ['<arquivo>']|off

seed: O comando `seed` permite a especificação de uma semente definida para o gerador de números pseudo-aleatório da $\mathcal{ECLE}$. Isso permite a repetição de experimentos idênticos e a avaliação de parâmetros sem que a seqüência pseudo-aleatória seja uma variável adicional aos experimentos. A sintaxe do comando `seed` é descrita a seguir, sendo que `semente` deve ser um número inteiro.

seed <semente>

select: Para executar o algoritmo evolutivo, a $\mathcal{ECLE}$ inicia com a seleção de um conjunto de indivíduos para compor a população inicial. É possível, por exemplo, realizar uma amostragem sem reposição de um determinado conjunto de regras e uma amostragem com reposição de um segundo conjunto de regras, desde que os conjuntos estejam em arquivos distintos. Duas sintaxes são permitidas:

select all|<num> [individuals] [with reposition|without reposition] [from '<file>']

select all|<num> [individuals] [from '<file>'] [with reposition| without reposition]

use: O comando `use` especifica quais métodos serão utilizados pelo algoritmo evolutivo. Este comando determina métodos de avaliação, seleção, *crossover* e mutação, além de especificar o uso do elitismo. Três sintaxes são permitidas:

`use <categoria> <método> [<lista de parâmetros>]`

`use <categoria> composite`

`use elitism`

Sendo que `categoria` determina a categoria de ferramentas evolutivas configuradas pelo comando. Atualmente, os quatro valores possíveis são **measure**, **selection**, ***crossover***, **mutation** e **iterator**. O comando `use` deve ser utilizado no máximo uma vez para cada categoria disponível.

add: Quando o método escolhido para alguma categoria for de composição (`composite`), é necessário definir todos os métodos componentes através do comando add. A sintaxe do comando add é:

`add <categoria> <método> [param <valor>] [<lista de parâmetros>]`

Sendo que `categoria`, `método` e `lista de parâmetros` têm o mesmo significado que no comando `use`.

stop: O comando `stop` pode ser utilizado para definir um ou mais critérios de parada. Se, a qualquer momento da execução, algum critério `stop` for verificado, a execução é finalizada, independente dos outros critérios. A sintaxe do comando stop é:

`stop [when] <atributo> <operador> <valor>`

Sendo que `atributo` é uma expressão lógica envolvendo os seguintes critérios: `iterations`, `time`, `convergence`, `best` ou `mean`. O conjunto de operadores disponível é {>, <, >=, <=}. Adicionalmente, quando `atributo` for `convergence`, o comando `stop` pode assumir a seguinte sintaxe:

`stop [when] convergence <operador> <valor> [with window <size>]`

Sendo que `size` é o tamanho do intervalo para o cálculo do desvio padrão para verificar a estabilidade da melhor solução (valor padrão igual a 20). Também, nesse caso, é possível definir o limiar de teste da convergência, cujo valor por padrão é igual 0.005.

Especificação de parâmetros: É importante observar que muitos métodos na $\mathcal{ECLE}$ exigem a especificação de parâmetros individuais. Por exemplo, o método de seleção por torneio necessita da especificação do tamanho

do torneio, enquanto que o *crossover* local necessita do parâmetro que informa a taxa de mistura (α). A lista de parâmetros do comando `use` especifica esses valores. A sintaxe dessa lista é:

```
with  <parâmetro 1> <valor 1>
      <parâmetro 2> <valor 2> and
      ...
      <parâmetro N> <valor N>
```

Diversos parâmetros são fornecidos pela $\mathcal{ECLE}$, cujo valor é uma lista de argumentos entre colchetes. Exemplos de uso da especificação de parâmetros são:

`use selection ExponentialRanking with base [0.7]:` especifica o método de seleção *ranking* exponencial e configura o parâmetro `base` com o valor `0.7`.

`use selection LinearRanking with range [0.10]:` especifica o método de seleção por *ranking* linear com intervalo de atuação [0, 10].

`use crossover LocalLevel with crossover_rate [0.8]:` especifica o método de *crossover* local com taxa de *crossover* `0.8`.

Inserção de Comentários: O arquivo de configuração permite a inclusão de comentários com o caractere especial `#`. Todo o restante da linha à direita deste caractere (a menos que ele apareça no nome de arquivo) será ignorado.

5.7 Um Exemplo de Configuração

Para ilustrar uma possível configuração da $\mathcal{ECLE}$ para executar o algoritmo evolutivo, segue a informação contida no arquivo de configuração correspondente. Maiores detalhes podem ser encontrados em Giusti et al. (2006).

```
# Experimento #7
# Date  : 03/16/2006
# Author: Adriano Donizete Pila / Rafael Giusti

# Seleciona aleatoriamente indivíduos gerados pelo
# classificador e incluí regras de nossa autoria
# do arquivo 'personal.rules'
```

```
select 26 individuals without reposition from 'c4.5.rules'
select all from 'personal.rules'

# Função fitness: f(x) = (2*Rank(Novidade)
#       + Rank(Laplace) + Rank(Suporte))/4


use measure CompositeMORanking with mean_type ['HarmonicMean']
add measure Novelty param [2]
add measure Laplace param [1]
add measure Suporte param [1]

# Usa seleção por torneio com tamanho do torneio 3.

use selection Tournament with tournament_size [3]

# Utiliza crossover composto com probabilidades idênticas
# para os níveis estrutural e de atributo (60%) e probabilidade
# de 80% para o crossover local


use crossover composite
add crossover StructuralLevel with crossover_rate [0.60]
add crossover FeatureLevel with crossover_rate [0.60]
add crossover LocalLevel with crossover_rate [0.80] and
alpha_rate [0.3]

# Utiliza mutação composta com baixa probabilidade (5%) para
# o nível estrutural e uma probabilidade maior (20%)
# para o nível local

use mutation composite
add mutation StructuralLevel with mutation_rate [0.05]
add mutation LocalLevel with mutation_rate [0.2]

# Para após uma hora de execução ou quando ocorrer
# convergência a contar da iteração número 10.
stop when time >= 3600
stop when convergence <= 0.01 with window 10
```

Nessa execução, a população inicial do algoritmo evolutivo será constituída por 26 regras selecionadas aleatoriamente e sem reposição do arquivo 'c4.5.rules' mais todas as regras no arquivo 'personal.rules', indicado pelos primeiros comandos **`select`**. Após, é definido a função a ser utilizada para avaliar os indivíduos. A $\mathcal{ECLE}$ fornece diversas funções de avaliação, as quais são reconhecidas pelo seu nome, **`CompositeMORanking`** neste exemplo. Essa função utiliza *rankings* para compor a medida final utilizando a média harmônica, e o usuário específica após quais as medidas a serem *rankeadas* e o peso de cada *rank* individual. Após indicar o uso do método de seleção por torneio, com tamanho de torneio igual a 3, é indicado que serão utilizados vários métodos de *crossover*, *i.e.* **`use composite crossover`**. No exemplo considerado, os três *crossovers* são utilizados: o *crossover* estrutural com taxa de 60%; o *crossover* em nível de atributo com taxa de 60%; e o *crossover* local com taxa de 80% e 0.3 de fator de mistura (α). Também são utilizados os dois métodos de mutação: estrutural com taxa de 5% e a local com taxa de 2%. Finalmente, é definido o critério de parada do algoritmo genético. No exemplo considerado, o algoritmo pará após uma hora de execução (3600 segundos) ou quando o desvio padrão da função de avaliação do melhor indivíduo nas 10 (*window*) últimas iterações for ≤ 0.01. A $\mathcal{ECLE}$ utiliza os valores padrões dos possíveis comandos, caso o usuário não defina explicitamente esses valores no arquivo de configuração.

5.8 O Ambiente Computacional SNIFFERECLE

Como mencionado, a avaliação experimental tem papel fundamental na verificação da adequabilidade de um algoritmo como o proposto neste trabalho. Dessa forma, para verificar a adequabilidade do algoritmo proposto, foi desenvolvido um ambiente capaz de realizar diversos experimentos computacionais utilizando a grande gama de parâmetros que a implementação de nosso algoritmo fornece. Esse ambiente é implementado como uma extensão do SNIFFER — Seção 5.3.2 — o qual tem-se mostrado extremamente apropriado para a execução organizada de diversos tipos de experimentos. O ambiente implementado neste trabalho como uma extensão do SNIFFER é chamado de SNIFFERECLE.

Para organizar um experimento, o ambiente computacional SNIFFER utiliza a estrutura de um sistema de arquivos do sistema operacional no qual está sendo executado. Os conjuntos de dados a serem utilizados são dispostos em diretórios, sendo que alguns identificadores de diretórios possuem signifi-

cado especial para o ambiente. Isso permite utilizar as facilidades do SNIFFER para definir novos identificadores que indiquem novas classes de experimentos, como no caso do SNIFFERECLE. Por exemplo, na Figura 5.6 é mostrado um experimento organizado pelo usuário que utiliza o SNIFFER e também o SNIFFERECLE.

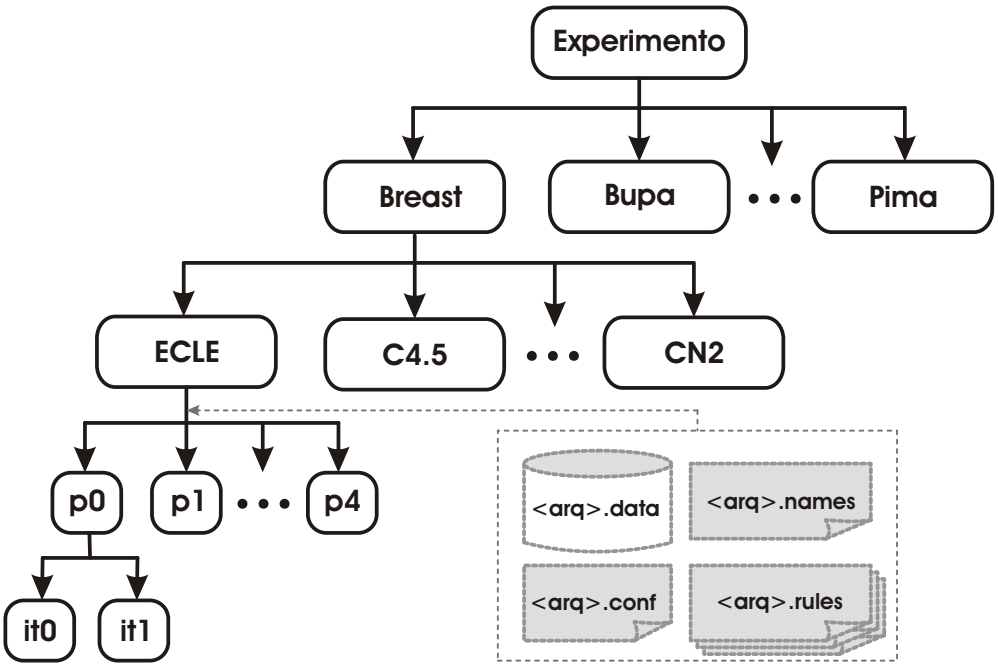

Figura 5.6: Esquema de funcionamento do SNIFFER e do SNIFFERECLE

Existe um diretório raiz para o experimento chamado `Experimento`. Dentro do diretório, o qual engloba todo o experimento, foram criados os diretórios `Breast`, `Bupa`, `...` e `Pima`, um para cada conjunto de dados a ser utilizado. Assim, cada um desses diretórios contem o conjunto de dados a ser utilizado. Dentro do diretório `Breast` foram criados os diretórios `ECLE`, `C4.5`, ... e `CN2`[2]. Os nomes desses diretórios constituem identificadores que possuem um significado especial para o SNIFFER. Por exemplo, o identificador `C4.5` indica que o experimento definido pelo usuário nesse diretório será realizado utilizando o algoritmo de aprendizado $\mathcal{C}4.5$, enquanto que o identificador `CN2` refere-se ao $\mathcal{CN}2$. O SNIFFER trabalha atualmente com aproximadamente dez diferentes algoritmos de aprendizado supervisionado. Detalhes sobre esses tipos de experimentos que utilizam algoritmos de aprendizado podem ser consultados em Batista (2003).

[2]Nessa figura são somente mostrados os diretórios contidos no diretório `Breast`, os demais foram omitidos por simplificação.

O SNIFFERECLE, implementado neste trabalho, é ativado pelo SNIFFER quando o identificador `ECLE` for o nome do diretório — Figura 5.6. O SNIFFERECLE é composto por duas classes que em conjunto são capazes de reconhecer as configurações fornecidas no arquivo de configuração e executar uma série de experimentos personalizados. Em termos de implementação, a classe **SnifferECLE** é uma especialização da classe **Sniffer** pertencente à DOL (Batista, 2003; Batista & Monard, 2003b).

Deve ser observado que os arquivos de configuração (<arq>.conf), o conjunto inicial de regras (<arq>.rules) e o conjunto de dados a ser utilizado, expresso na sintaxe DSX do DISCOVER[3], (<arq>.data e <arq>.names) devem estar armazenados no diretório de nome `ECLE`.

Uma vez que o SNIFFERECLE tenha encontrado os 4 (quatro) arquivos necessários, é feita uma validação do tipo *5x2-cross-validation*, *i.e.* são criados 5 *folds* (diretórios `p0..p4`) e para cada *fold* o conjunto de dados é dividido em duas partições, sendo ambas armazenadas nos diretórios `it0` e `it1`. A diferença entre `it0` e `it1` é que neste último a partição utilizada como conjunto de treinamento em `it0` passa a ser o conjunto de teste e a partição utilizada como conjunto de teste em `it0` passa a ser o conjunto de treinamento. Para cada iteração é feita uma chamada simples da $\mathcal{ECLE}$, sendo que os parâmetros de configuração do algoritmo evolutivo são aqueles fornecidos pelo arquivo de configuração. Se houver mais de um experimento diferente a ser realizado com a $\mathcal{ECLE}$, cada um deve ser mantido no seu próprio diretório `ECLE`.

Na Figura 5.7 é ilustrado como é executado um dos experimentos descritos no próximo capítulo utilizando o conjunto de dados breast e 8 (oito) configurações diferentes, onde **Confi**, refere-se aos arquivos `<arq>.data`, `<arq>.names`, `<arq>.rules` e `<arq>.conf` a serem utilizados na execução desse experimento específico. Na realidade, nada impede organizar a totalidade dos experimentos realizados descritos no próximo capítulo, como um experimento único, colocando no primeiro nível todos os conjuntos de dados considerados e não somente um conjunto de dados. Também, é importante salientar que a maneira que os experimentos encontram-se organizados utilizando um sistema de diretórios, permite disparar esses processos para serem executados em paralelo.

Além das facilidades na execução dos experimentos, o SNIFFERECLE também faz a coleta dos resultados de todos os *folds*, gerando informações consolidadas que são armazenadas em arquivos texto. Com essas mesmas informações, são pelo geradas pelo SNIFFERECLE tabelas no formato LaTeX, como as presen-

[3]Existem facilidades já implementadas para transformar conjuntos de dados expressos em diversas sintaxes para a sintaxe DSX do DISCOVER.

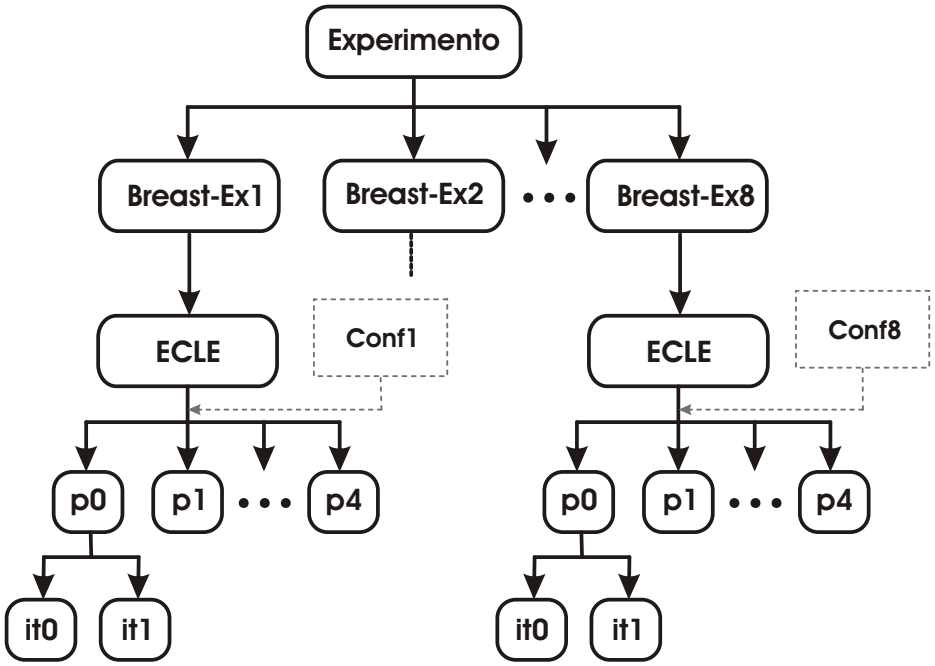

Figura 5.7: Exemplo funcionamento do SNIFFERECLE

tes no Apêndice A na página 133. Mais uma vez salientamos a importância de um ambiente computacional para automatizar, tanto quanto possível, a realização de experimentos, os quais são fundamentais nesta área de pesquisa.

5.9 Considerações Finais

Neste capítulo foi apresentada a arquitetura geral da $\mathcal{ECLE}$ que é a implementação do algoritmo evolutivo e das idéias propostas no Capítulo 4. Foi brevemente apresentada a estrutura de classes que compõem a $\mathcal{ECLE}$ e como essa biblioteca de classes se relaciona com algumas outras bibliotecas de classes já existentes dentro do DISCOVER. Cada uma das classes abstratas que fornecem métodos primitivos para as classes especializadas foi sucintamente apresentada. Para executar experimentos utilizando o algoritmo evolutivo proposto, foi também implementado um ambiente, o SNIFFERECLE, que é capaz de gerenciar uma vasta gama de experimentos utilizando as informações do arquivo de configuração cuja gramática, por nós proposta, é de fácil compreensão, não necessitando, portanto, que o usuário chame explicitamente pelas funcionalidades da $\mathcal{ECLE}$. No próximo capítulo é apresentada a avaliação experimental da $\mathcal{ECLE}$ utilizando diversos conjuntos de dados.

Capítulo

6

AVALIAÇÃO EXPERIMENTAL

Para cada coisa que acredito saber, dou-me conta de nove que ignoro.
Provérbio Árabe

6.1 Considerações Iniciais

NESTE capítulo é apresentada uma avaliação experimental da nossa proposta com o objetivo de comprovar sua adequabilidade na construção de regras de conhecimento com propriedades especificas. Para realizar a avaliação, foram selecionados diversos conjuntos de dados com diferentes características. Também, as regras de conhecimento iniciais (indivíduos) que constituem a população inicial do algoritmo evolutivo foram obtidas por meio de amostragem de regras criadas por algoritmos de aprendizado simbólico e de regras de associação. O comportamento do algoritmo evolutivo foi avaliado utilizando funções de avaliação simples-objetivo e a função multi-objetivo baseada em *ranking*, bem como diversos cenários que estão relacionados à diferentes métodos de seleção, *crossover* e mutação.

6.2 Conjuntos de Dados

Os conjuntos de dados utilizados para a realização dos experimentos apresentados a seguir consistem de conjuntos de dados naturais selecionados do repositório de dados da UCI (Newman et al., 1998). A diversidade existente entre os conjuntos de dados foi a linha guia na escolha dos mesmos, com o objetivo de verificar a adequabilidade da nossa proposta em conjuntos de dados com características diferentes, tais como: número de atributos, número de exemplos, número de atributos categóricos/numéricos e número de classes. No final desse processo foram selecionados 7 (sete) conjuntos de dados supervisionados pouco desbalanceados, a fim de não introduzir interferências associadas ao uso de um ou outro método para tratar o problema de classes desbalanceadas (Batista et al., 2004a). O domínio de cada conjunto de dados é brevemente descrito a seguir.

Breast: o problema é predizer se uma amostra de tecido de mama obtida de uma paciente é maligna ou benigna baseada em dados histológicos.

Cmc: o objetivo é predizer o método contraceptivo utilizado com base nos dados sócio-culturais do casal e nos dados demográficos da região em que vivem.

Heart: o problema é determinar a presença de doença cardíaca com base em informações colhidas do paciente em exames de rotina que aferem a pressão sanguínea, taxa de colesterol, taxa de glicemia e desempenho em exercícios físicos.

Nursery: a meta é predizer a ordem de prioridade em cursar uma escola de enfermagem com base nos dados sócio-financeiros da família.

Spambase: o objetivo é classificar mensagens eletrônicas como sendo ou não SPAM com base na freqüência de ocorrência de determinadas palavras no texto das mensagens.

Vehicle: o objetivo é classificar tipos de veículos, usando um conjunto de atributos extraídos a partir de suas silhuetas. O veículo pode ser visto de diversos ângulos.

Wbc: a meta é predizer se determinada amostra de tecido apresenta cancer maligno ou benigno. Embora o objetivo seja o mesmo do conjunto Breast, os atributos neste conjunto de dados são diferentes do anterior. Além disso, os exemplos foram colhidos em outra localidade (Hospital da Universidade de Wisconsin) e com outros pacientes.

Na Tabela 6.1 são apresentadas as características originais dos conjuntos de dados utilizados nos experimentos, as quais foram obtidas utilizando ferramentas desenvolvidas no trabalho de Voltolini (2006). Os conjuntos de dados foram previamente pré-processados e os exemplos com valores faltantes de atributos foram suprimidos. A principal motivação para retirar os exemplos com essas características deve-se ao fato de que o tratamento de exemplos com valores ausentes representa outra variável no problema e exige um tratamento específico (Batista, 2003). Dessa forma, as informações presentes na Tabela 6.2 são relativas aos conjuntos de dados contendo apenas exemplos para os quais todos os valores dos atributos são conhecidos[1].

Conj. dados	# Ex	# Dupl	# Classes	Classes (% Ex)	Erro C.M.	# Atrib (nom,num)
breast	285	2	2	no_recurrence 70,53 recurrence 29,47	29,47	9(5,4)
cmc	1473	48	3	1 42,70 2 22,61 3 34,69	57,30	9(0,9)
heart	270	0	2	1 55,56 2 44,44	44,44	13(0,13)
nursery	12960	0	5	not_recom 33,33 priority 32,92 recommend 0,02 spec_prior 31,20 very_recom 2,53	66,67	8(8,0)
spambase	4601	391	2	0 60,60 1 39,40	39,40	57(0,57)
vehicle	846	0	4	bus 25,77 opel 25,06 saab 25,65 van 23,52	74,23	18(0,18)
wbc	699	8	2	2 65,52 4 34,48	34,48	10(0,10)

Tabela 6.1: Características dos conjuntos de dados — original

Como pode ser observado na Tabela 6.2, os conjuntos de dados escolhidos possuem características diferentes quando comparados uns com os outros, apresentando variações no número de exemplos, números de classes, número de atributos nominais/numéricos e número de exemplos duplicados.

[1]Somente os conjuntos de dados originais breast e wbc apresentam atributos com valores desconhecidos.

Conj. dados	# Ex	# Dupl	# Classes	Classes (% Ex)	Erro C.M.	# Atrib (nom,num)
breast	276	2	2	no_recurrence 71,01 recurrence 28,99	28,99	9(5,4)
cmc	1473	48	3	1 42,70 2 22,61 3 34,69	57,30	9(0,9)
heart	270	0	2	1 55,56 2 44,44	44,44	13(0,13)
nursery	12960	0	5	not_recom 33,33 priority 32,92 recommend 0,02 spec_prior 31,20 very_recom 2,53	66,67	8(8,0)
spambase	4601	391	2	0 60,60 1 39,40	39,40	57(0,57)
vehicle	846	0	4	bus 25,77 opel 25,06 saab 25,65 van 23,52	74,23	18(0,18)
wbc	683	8	2	2 65,01 4 34,99	34,99	10(0,10)

Tabela 6.2: Características dos conjuntos de dados — sem valores ausentes

6.3 Base de Regras

Como mencionado, o conjunto de regras de conhecimento que constituem a população inicial do algoritmo evolutivo pode ser constituído por regras geradas por algoritmos de aprendizado simbólico, por outros algoritmos ou fornecidas pelo próprio usuário. Neste trabalho, esse conjunto de regras, relacionado a cada um dos conjuntos de dados, foi obtido utilizando os indutores simbólicos $\mathcal{CN}2$ (Clark & Niblett, 1989) e $\mathcal{C}4.5$ (Quinlan, 1988), bem como o gerador de regras de associação $\mathcal{APRIORI}$ (Agrawal & Srikant, 1998). Ambos indutores foram executados com os valores *default* dos parâmetros, enquanto que para o gerador de regras de associação $\mathcal{APRIORI}$ foi elevado o suporte mínimo à 70% para evitar a geração de muitas regras que cubram poucos exemplos (o valor *default* é 10%). Utilizando todos os exemplos de cada conjunto de dados, foram geradas as regras de conhecimento respectivas. Na Tabela 6.3 é mostrado o número total de regras geradas por cada um dos algoritmos utilizado.

Conj. Dados	# Ex	# Atrib	# $\mathcal{CN}2$	# $\mathcal{C}4.5$	# $\mathcal{APRIORI}$
breast	276	9	38	28	156
cmc	1473	9	174	150	10
heart	270	13	28	27	4403
nursery	12961	8	329	359	290
spambase	4601	57	115	119	49080
vehicle	846	18	50	98	52640
wbc	683	10	12	15	3169

Tabela 6.3: Número total de regras para cada conjunto de dados

Desse conjunto total de regras, foram selecionadas aleatoriamente N_{Regras} para compor a população inicial da $\mathcal{ECLE}$. Para cada conjunto de dados, o valor de N_{Regras} é dado pela Equação 6.1, a qual leva em consideração o número de exemplos e o número de atributos do conjunto de dados. A Equação 6.1 foi obtida de forma empírica, como forma de obter um número de regras que contenha um bom equilíbrio entre o número de exemplos e o número de atributos.

$$N_{Regras} = log_2(\#Ex) \times \sqrt{\#Atrib} \tag{6.1}$$

Essas N_{Regras} iniciais foram selecionadas aleatoriamente com a seguinte distribuição: 40% de regras proveniente do $\mathcal{CN}2$, 40% de regras proveniente do $\mathcal{C}4.5$ e 20% de regras proveniente do $\mathcal{APRIORI}$. Utilizou-se uma proporção superior para as regras provenientes dos indutores $\mathcal{CN}2$ e $\mathcal{C}4.5$ pois as regras do $\mathcal{APRIORI}$ são utilizadas com o objetivo de inserir diversidade no conjunto inicial de regras. Na Tabela 6.4 é mostrado o número de regras iniciais (indivíduos) para cada um dos conjuntos de dados, bem como o número de regras provenientes de cada um dos algoritmos que geraram essas regras.

Conj. Dados	# Ex	# Atrib	# $\mathcal{CN}2$	# $\mathcal{C}4.5$	# $\mathcal{APRIORI}$	N_{Regras}
breast	276	9	10	10	5	24
cmc	1473	9	13	13	6	32
heart	270	13	12	12	6	29
nursery	12961	8	15	15	8	39
spambase	4601	57	37	37	18	92
vehicle	846	18	17	17	8	41
wbc	683	10	12	12	6	30

Tabela 6.4: Número inicial de regras para cada conjunto de dados

6.4 Funções de Avaliação

Para executar os experimentos foram escolhidas 5 (cinco) medidas de avaliação de regras para comporem as diferentes funções de avaliação. Na Tabela 6.5 são descritas as funções de avaliação e as medidas de avaliação de regras que foram utilizadas para compor cada função. As 5 (cinco) medidas foram individualmente utilizadas como função de avaliação simples-objetivo (*Fnc1a*, *Fnc1b*, *Fnc1c*, *Fnc2a* e *Fnc2b*), bem como foram combinadas utilizando *ranking* para compor a função de avaliação multi-objetivo (*Fnc1* e *Fnc2*). Nesse último caso, não foram atribuídos pesos aos *ranks* das medidas de forma que alguma fosse privilegiada na combinação. As medidas individuais foram escolhidas levando em conta que as duas funções de avaliação multi-objetivo

que utilizam um subconjunto dessas cinco medidas individuais apresentam características interessantes para encontrar regras de conhecimento.

Função de Avaliação	Medida(s) de Avaliação de Regras Utilizada(s)	Equação referência
Fnc1a	Laplace	Equação 2.2 na página 14
Fnc1b	Suporte	Equação 2.8 na página 15
Fnc1c	Novidade	Equação 2.9 na página 15
Fnc2a	Sensibilidade	Equação 2.5 na página 14
Fnc2b	Especificidade	Equação 2.6 na página 15
Fnc1	Laplace Suporte Novidade	
Fnc2	Sensibilidade Especificidade	

Tabela 6.5: Resumo das funções de avaliação utilizadas

6.5 Cenários

Como descrito previamente no Capítulo 4, o algoritmo evolutivo proposto admite diferentes métodos de seleção, *crossover* e mutação, os quais encontram-se implementados na biblioteca $\mathcal{ECLE}$. Assim, existe uma gama grande de possibilidades de combinações desses métodos e seus parâmetros. Para efeito de comparação de algumas dessas combinações e para analisar o resultado de algumas combinações específicas, optou-se por 8 (oito) combinações, chamadas de agora em diante de cenários, os quais encontram-se descritos de forma resumida na Tabela6.6.

Na definição dos cenários procurou-se combinações de parâmetros diferentes que pudessem explorar e verificar o desempenho do algoritmo evolutivo proposto utilizando diversos métodos de seleção e métodos e taxas de *crossover* e mutação. Para tanto, foram escolhidos 2 (dois) métodos de seleção: roda da roleta, cuja probabilidade de seleção é proporcional à aptidão dos indivíduos (Seção 3.5.2.1 na página 35) e torneio, cuja seleção é feita pelo sorteio aleatório de um número fixo de elementos e dentre esses o de maior aptidão é o vencedor (Seção 3.5.2.2 na página 36). No caso da seleção por torneio, o tamanho foi fixado em 3 (três), que é o número de elementos aleatoriamente sorteado da população para eleger o vencedor. Esse valor foi mantido fixo porque não é objetivo da avaliação estudar sua influência no resultado dos experimentos. Ainda, a escolha do valor 3 (três) reflete um padrão adotado na maioria dos algoritmos evolutivos que utilizam esse método de seleção.

No caso dos operadores de *crossover*, os quais são aplicados em pais que predizem a mesma classe, foram utilizados os três métodos propostos e implementados na $\mathcal{ECLE}$: estrutural (recombina entre as condições), atributo (recombina

Cenário	Método de Seleção	Métodos de *Crossover*	(Taxa%)	Métodos de Mutação	(Taxa%)
Cnr1	Roda da Roleta	Estrutural	(80%)	Estrutural	(40%)
		Atributo	(80%)	Local	(40%)
		Local (com $\alpha = 0.3$)	(80%)		
Cnr2	Roda da Roleta	Estrutural	(80%)	Estrutural	(5%)
		Atributo	(80%)	Local	(5%)
		Local (com $\alpha = 0.3$)	(80%)		
Cnr3	Roda da Roleta	Estrutural	(20%)	Estrutural	(40%)
		Atributo	(20%)	Local	(40%)
		Local (com $\alpha = 0.3$)	(20%)		
Cnr4	Roda da Roleta	Estrutural	(20%)	Estrutural	(5%)
		Atributo	(20%)	Local	(5%)
		Local (com $\alpha = 0.3$)	(20%)		
Cnr5	Torneio	Estrutural	(80%)	Estrutural	(40%)
	(#torneio = 3)	Atributo	(80%)	Local	(40%)
		Local (com $\alpha = 0.3$)	(80%)		
Cnr6	Torneio	Estrutural	(80%)	Estrutural	(5%)
	(#torneio = 3)	Atributo	(80%)	Local	(5%)
		Local (com $\alpha = 0.3$)	(80%)		
Cnr7	Torneio	Estrutural	(20%)	Estrutural	(40%)
	(#torneio = 3)	Atributo	(20%)	Local	(40%)
		Local (com $\alpha = 0.3$)	(20%)		
Cnr8	Torneio	Estrutural	(20%)	Estrutural	(5%)
	(#torneio = 3)	Atributo	(20%)	Local	(5%)
		Local (com $\alpha = 0.3$)	(20%)		

Tabela 6.6: Resumos dos cenários utilizados

entre os limitantes das condições) e local (recombina os valores das condições). Neste último caso, deve ser definida a taxa de mistura (α) que é o percentual da troca de carga genética entre os valores escolhidos para o *crossover* (Seção 4.6 na página 51). Esse valor foi fixado em 0.3 para que a mistura entre os valores dos atributos numéricos não fosse muito grande. Cada método tem sua probabilidade de ocorrência independente, ou seja, em um indivíduo que foi aplicado o operador de *crossover* estrutural, será aplicado outro *crossover* se a probabilidade de ocorrência desse outro for satisfeita. Como as probabilidades são independentes, taxas de *crossover* específicas podem ser adotadas. Entretanto, para não favorecer a ocorrência de determinado método, nos experimentos foi sempre utilizada a mesma taxa para os três métodos. Assim, foram escolhidas duas taxas de *crossover* para a configuração dos cenários: 20% e 80%. Com 80% de taxa de *crossover* espera-se que o método faça uma busca mais rápida, enquanto que com 20% de taxa de *crossover* espera-se que o método faça uma melhor exploração do espaço de busca. Uma vez que a taxa tenha sido fixada, a mesma é aplicada para os três tipos de métodos de *crossover*.

No caso dos operadores de mutação foram utilizados os dois operadores propostos e implementados na $\mathcal{ECLE}$: estrutural (muda a presença da condição) e local (muda o valor presente na condição). Novamente, como nos operadores de *crossover*, a probabilidade de ocorrência é independente e foi mantida a mesma taxa para ambos para não ocasionar qualquer favorecimento entre os métodos. Foram escolhidas duas taxas de mutação para a configuração dos

cenários: 5% e 40%. A taxa de 5% é a mais conservadora e insere pouca diversidade na população, fazendo pouca exploração no espaço de busca. Já a taxa de 40% é uma taxa bastante agressiva, possibilitando uma grande exploração do espaço de busca. Entretanto, pode ser que uma taxa alta de mutação degrade as boas partes dos indivíduos, o que prejudica a convergência do algoritmo evolutivo.

Todos esses parâmetros escolhidos foram combinados para originar os cenários presentes na Tabela 6.6 e, assim, verificar a influência desses parâmetros nos resultados do algoritmo evolutivo quando executado com os métodos e taxas de cada um desses cenários nos conjuntos de dados considerados.

6.6 Descrição dos Experimentos

Além dos 8 (oito) cenários, os experimentos realizados foram divididos em dois grupos — Grupo 1 e 2. Para ambos grupos o conjunto de regras iniciais do algoritmo evolutivo, *i.e.* a população inicial, foi criado como explicado na Seção 6.3 na página 90. Entretanto, no Grupo 1, para cada conjunto de dados o mesmo conjunto de regras foi utilizado em todos os experimentos com o objetivo de analisar o comportamento do algoritmo evolutivo nos diferentes cenários utilizando a mesma população inicial. No segundo caso — Grupo 2 — um conjunto de regras iniciais foi criado para cada cenário. Ou seja, todos os experimentos do Grupo 2 foram executados utilizando populações iniciais diferentes. O principal objetivo desse segundo grupo de experimentos consiste em verificar se o algoritmo evolutivo é capaz de construir boas regras com funções multi-objetivo, independentemente do conjunto de regras iniciais.

Todos os experimentos foram realizados utilizando *5x2-cross-validation* (Dietterich, 1998). Ou seja, após construída a população inicial, o algoritmo evolutivo é executado 5 (cinco) vezes, realizando em cada uma dessas execuções um processo de *2-cross-validation*, dividindo o conjunto de dados em duas partições diferentes com 50% dos exemplos em cada partição. Uma dessas partições é usada como conjunto de treinamento e a outra como conjunto de teste. A segunda execução é feita invertendo os papéis dos conjuntos de treinamento e teste. Esse processo é repetido cinco vezes, obtendo-se 10 (dez) execuções do experimento considerado, tal que os valores médios e o desvio padrão das medidas de interesse relacionadas ao algoritmo evolutivo podem ser calculados.

No Apêndice A na página 133 encontram-se tabelados, respectivamente, os re-

sultados experimentais correspondentes ao Grupo 1 (Tabelas A.1 — A.7) e ao Grupo 2 (Tabelas A.8 e A.9) para os oito cenários — Tabela 6.6 — utilizando as funções de avaliação descritas na Tabela 6.5. Nessas tabelas no Apêndice A, para cada conjunto de dados e cada um dos cenários são apresentadas duas linhas. Na primeira linha são apresentados o valor médio da função de avaliação na população inicial seguido do valor médio dessa função de avaliação na população final, após a convergência do algoritmo evolutivo. O valor entre parênteses refere-se à variação percentual média entre o valor inicial e final da função de avaliação em cada um dos 10-*folds*. Para o mesmo cenário, na segunda linha, são apresentados os respectivos desvios padrão dessas médias. Na coluna rotulada $+/-$ é apresentado o número de vezes que o valor final da função de avaliação melhorou em cada um dos 10-*folds* ($+$), bem como quantas vezes esse valor diminuiu ($-$). No caso da função de avaliação ser multi-objetivo (*Fnc1* e *Fnc2* na Tabela 6.5), a contagem em cada *fold* é $+1$ se a maioria das medidas que participam da função de avaliação multi-objetivo melhoraram no *fold*, caso contrário a contagem é -1. Finalmente, na coluna rotulada #Iterações, é apresentado o valor inteiro que aproxima o número médio de iterações até a convergência do algoritmo evolutivo e, logo abaixo, o valor inteiro que aproxima o desvio padrão do número médio de iterações.

6.7 Resultados Experimentais — Grupo 1

Como mencionado, todos os experimentos do Grupo 1 foram realizados executando o algoritmo evolutivo com a mesma população inicial (regras), com o objetivo de verificar a influência dos diversos cenários, *i.e.* a influência dos diferentes valores dos parâmetros utilizados para executar o algoritmo evolutivo. Inicialmente são apresentados os resultados considerando cada uma das cinco medidas de regras como função de avaliação (função de avaliação simples-objetivo) e após os resultados considerando as funções de avaliação multi-objetivo.

6.7.1 Funções de Avaliação Simples-Objetivo

Nas Figuras B.1 à B.7 na página 157 são ilustradas graficamente as informações das Tabelas A.1 à A.5 na página 143. Em cada figura é mostrado, para cada conjunto de dados, cenário e utilizando como função de avaliação somente uma medida de regra (simples-objetivo) a relação entre os valores inicial e final dessas medidas. No eixo das abscissas encontra-se representado o

valor médio inicial da medida (função simples-objetivo) — f_i — e no eixo das ordenadas o valor médio final — f_f — atingido pelo algoritmo evolutivo. Ou seja, se o ponto (f_i, f_f) estiver acima da reta $y = x$, então $f_f > f_i$. Para cada ponto que representa o valor inicial e final do valor médio da medida no respectivo cenário, é também informado o número de vezes que o valor final da função de avaliação melhorou (+) e/ou diminuiu (−) em cada um dos 10 *folds*.

Analisando essas figuras é possível observar que para a medida de especificidade, o comportamento é o mesmo para todos os conjuntos de dados e cenários. Em todos os casos a população inicial de regras (indivíduos) contém pelo menos uma regra com máximo valor de especificidade (especificidade=1) e o algoritmo evolutivo converge no menor número possível de iterações (intervalo=21). Analisando os *folds*, foi possível observar que esse valor ótimo foi mantido em quase todos os *folds*, devido ao operador de elitismo que copia o melhor indivíduo de uma geração para outra.

Para exemplificar, a Regra 6.1 mostra a melhor regra inicial do cenário *Cnr1*, para o conjunto de dados cmc, extraída de um dos *folds*[2]. A especificilidade dessa regra é 1 (dada por $\frac{\overline{bh}}{\overline{h}}$ ou pelo seu equivalente $\frac{f_{\overline{bh}}}{f_{\overline{h}}}$), já que $\overline{h} = 571$ ($\overline{h} = \overline{bh} + b\overline{h}$ ou $f_{\overline{h}} = f_{\overline{bh}} + f_{b\overline{h}}$). A Regra 6.2 é a melhor regra final construída pelo algoritmo evolutivo, também com especificidade=1, mas trata-se de uma regra diferente da regra inicial, e diferente do conjunto de regras (indivíduos) na população inicial.

IF **hedu** = 4
AND **hocu** = 3
AND **medexp** = 0
AND **nchi** $>$ 2
AND **nchi** $\leq$ 16
AND **wage** > 41
AND **wage** $\leq$ 46
AND **wedu** = 4
AND **work** = 1
THEN **CLASS** = *2*

bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
0,	0,	571,	166,	737

[0.0000, 0.0000, 0.7748, 0.2252, 737]

Regra 6.1: Exemplo de regra inicial – cmc – Especificidade

Ainda que ambas regras possuam especificidade máxima, deve ser observado que a primeira regra especifica que a totalidade dos exemplos (737) não é da classe 2, sendo que isso é verdade somente para 571 dos exemplos ($Tn = \overline{bh}$). A situação é semelhante para a regra final, para a qual somente 422 dos

[2]Observe que a matriz de contingência para exemplos com valores faltantes não é apresentada na descrição das regras (todos os elementos são zero). Também, ainda que a $\mathcal{ECLE}$ somente utiliza a matriz de contingência expressa como freqüências relativas —Tabela 2.4 — para facilitar a interpretação das regras, foram incluídos os nomes dos elementos da matriz de contingência, bem como o valor absoluto dos mesmos, como descrito na Tabela 2.3 na página 13.

IF **hocu** = 2
AND **medexp** = 1
AND **nchi** $>$ 0
AND **nchi** $\leq$ 12
AND **wage** $>$ 19.09
AND **wage** $<$ 35.7
AND **wedu** = 4
AND **stdliv** = 4
THEN **CLASS** = *1*

bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
0,	0,	422,	315,	737
[0.0000,	0.0000,	0.5726,	0.4274,	737]

Regra 6.2: Exemplo de regra final – cmc – Especificidade

exemplos não são da classe 2. Funções multi-objetivo permitem especificar características que incluam mais de uma medida, o que permite evitar a construção desse tipo de regras.

É importante observar que os valores das matrizes de contingência das regras finais evoluídas mostrados nos exemplos, referem-se aos valores calculados durante a execução do algoritmo evolutivo, *i.e.* utilizando o conjunto de treinamento, enquanto que os valores utilizados para calcular os valores médios da medida são calculados sobre o conjunto de teste. No caso da especificidade, pode ser observado na Tabela A.5 que a regra evoluída também possui especificidade máxima quando calculada sobre o conjunto de teste, melhorando em todos os *folds* para todos os conjuntos de dados (+10).

Para a medida de Laplace, verifica-se que não houve grandes variações nos diferentes cenários. Os valores finais em todos os cenários para todos os conjuntos de dados ficaram muito próximos em relação aos valores iniciais. É possível notar que a melhor regra inicial do algoritmo evolutivo, exceto para o conjunto de dados cmc, tem valor dessa medida próximo à 1. Para o conjunto de dados cmc — Figura B.2 na página 154 — o valor da medida de Laplace da melhor regra inicial está mais distante de 1, mas o comportamento do algoritmo evolutivo é também semelhante nos diferentes cenários. É importante observar que os algoritmos $\mathcal{CN}2$ e $\mathcal{C}4.5$, dos quais são utilizadas 80% das regras por eles induzidas para construir a população inicial, tentam maximizar essa medida, que favorece regras que cobrem mais exemplos. Assim, não há muito para ser melhorado pelo algoritmo evolutivo. Isso pode ser também observado na grande variação dos valores de ganho/perda (+/-), exceto para o conjunto de dados nursery.

Para exemplificar, a Regra 6.3 é a melhor regra final de um dos *folds* do cenário *Cnr1* do conjunto de dados nursery. A Regra 6.4 é a regra final construída pelo algoritmo evolutivo. Os valores da medida de Laplace são 0.9960 e 0.9986, respectivamente, uma diferença mínima na variação. Entretanto, existe uma diferença grande no número de exemplos corretamente cobertos (bh), com am-

pla vantagem para a Regra 6.4. A regra inicial cobre corretamente 750 dos 2160 exemplos da classe da regra, enquanto que a regra final cobre corretamente todos os exemplos positivos (classe *not_recom*) e não cobre nenhum exemplo da classe negativa (os exemplos com classe diferentes de *not_recom*).

		bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
IF	**health** = not_recomended					
	AND **parents** = usual	750,	0,	4320,	1410,	6480
THEN	**CLASS** = *not_recom*	[0.1157,	0.0000,	0.6667,	0.2176,	6480]

Regra 6.3: Exemplo de regra inicial – nursery – Laplace

		bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
IF	**health** = not_recomended					
		2160,	0,	4320,	0,	6480
THEN	**CLASS** = *not_recom*	[0.333,	0.0000,	0.6667,	0.0000,	6480]

Regra 6.4: Exemplo de regra final – nursery – Laplace

No caso da medida de suporte, número relativo de exemplos cobertos corretamente pela regra (f_{hb}), o comportamento do algoritmo evolutivo é semelhante para todos os conjuntos de dados e cenários. Em todos os casos, como esperado, a regra final evoluída tem suporte semelhante ao percentual de exemplos — Tabela 6.2 na página 90 — que tem a mesma classe que a regra evoluída. Por exemplo, para o conjunto breast, a regra final de um dos folds é dada pela Regra 6.5, onde $f_{hb} = 0.7101$, igual à proporção de exemplos da classe *no_recurrence* no conjunto de exemplos.

		bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
IF	**involved_nodes** ≥ 0					
		98,	40,	0,	0,	138
THEN	**CLASS** = *no_recurrence*	[0.7101,	0.2899,	0.0000,	0.0000,	138]

Regra 6.5: Exemplo de regra final – breast – Suporte

Para a medida de novidade, o comportamento é também semelhante em todos os cenários e conjuntos de dados. Exceto nos conjuntos de dados nursery e spambase, nos outros conjuntos essa medida melhorou muito pouco e não consegue atingir seu máximo valor de 0.25. Por exemplo, a Regra 6.6 mostra a melhor regra inicial presente em um dos *folds* do conjunto de dados breast (novidade=0.0834) enquanto que a Regra 6.7 é a regra final evoluída (novi-

		bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
IF	**involved_nodes** ≤ 2.5					
	AND **irradiation** = no	74,	14,	26,	24,	138
THEN	**CLASS** = *no_recurrence*	[0.5362,	0.1014,	0.1884,	0.1739,	138]

Regra 6.6: Exemplo de regra inicial – breast – Novidade

dade=0.0886). Como pode ser observado, as matrizes de contingência diferem

somente quanto à cobertura de um exemplo. Também, embora a regra construída seja sintaticamente mais complexa, o número de exemplos cobertos corretamente (hb) é o mesmo.

IF	**age** ≥ 33					
	AND **age** ≤ 74					
	AND **degree_of_malig** ≥ 1					
	AND **degree_of_malig** ≤ 3					
	AND **involved_nodes** ≥ 0					
	AND **involved_nodes** ≤ 3.1	bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
	AND **irradiation** = no	74,	13,	27,	24,	138
THEN	**CLASS** = *no_recurrence*	[0.5362,	0.0942,	0.1957,	0.1739,	138]

Regra 6.7: Exemplo de regra final – breast – Novidade

Outro exemplo, a Regra 6.8 é a melhor regra inicial presente em um dos *folds* do conjunto de dados spambase (novidade=0.1234) enquanto que a Regra 6.9 é a regra final evoluída (novidade=0.1809). Pode ser observado que o algo-

IF	**char_freq_dolar_simbol** > 0					
	AND **char_freq_exclamation_point** > 0					
	AND **word_freq_hp** < 1	bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
	AND **word_freq_will** < 2	485,	26,	1368,	421,	2300
THEN	**CLASS** = *1*	[0.2109,	0.0113,	0.5948,	0.1830,	2300]

Regra 6.8: Exemplo de regra inicial – spambase – Novidade

IF	**char_freq_dolar_simbol** ≤ 0.05					
	AND **char_freq_exclamation_point** ≤ 0.38					
	AND **word_freq_000** ≤ 0.2					
	AND **word_freq_3d** ≤ 42.8					
	AND **word_freq_415** ≥ 0					
	AND **word_freq_addresses** ≥ 0					
	AND **word_freq_font** ≤ 0.12					
	AND **word_freq_free** ≤ 6.1					
	AND **word_freq_internet** ≤ 1.56					
	AND **word_freq_lab** ≥ 0					
	AND **word_freq_meeting** ≥ 0					
	AND **word_freq_money** ≤ 0.03					
	AND **word_freq_parts** ≥ 0					
	AND **word_freq_remove** ≤ 0	bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
	AND **word_freq_telnet** ≥ 0	1193,	89,	817,	201,	2300
THEN	**CLASS** = *0*	[0.5187,	0.0387,	0.3552,	0.0874,	2300]

Regra 6.9: Exemplo de regra final – spambase – Novidade

ritmo evolutivo, ao tentar obter uma regra com melhor novidade, optou por construir uma regra que prediz uma classe diferente da regra inicial. Uma outra observação quanto à Regra 6.9, está relacionada às condições que nela aparecem, considerando o domínio do conjunto de dados. Todos os valores dos atributos de spambase referem-se à freqüência de ocorrência de determinadas palavras no texto de mensagens eletrônicas. Assim, as condições

word_freq_415 ≥ 0; **word_freq_addresses** ≥ 0; **word_freq_lab** ≥ 0; **word_freq_meeting** ≥ 0; **word_freq_parts** ≥ 0; **word_freq_remove** ≤ 0 e **word_freq_telnet** ≥ 0, podem ser removidas dessa regra, já que essas condições são sempre verdadeiras. Porém, esse tipo de simplificação está estritamente ligada ao domínio considerado. Assim, é conveniente que seja realizada pelo especialista do domínio.

Para a medida de sensibilidade, que indica o número relativo de exemplos cobertos por uma regra que prevê a classe desses exemplo ($\frac{f_{hb}}{f_h}$), o algoritmo evolutivo consegue achar a regra final com sensibilidade próxima ao valor ótimo 1, independentemente da valor dessa medida na regra inicial, exceto para o conjunto de dados breast, nos cenários *Cnr2*, *Cnr4* e *Cnr6* — Figura B.1 na página 154 — e para o conjunto de dados cmc no cenários *Cnr4* — Figura B.2. Por exemplo, no conjunto de dados breast, a Regra 6.10 é a melhor regra inicial de um dos *folds*, com valor de sensibilidade 0.6039 e a Regra 6.11 a final com sensibilidade igual a 1. Ou seja, essa regra cobre todos os exemplos da classe *no_recurrence* no *fold*. Esse incremento na sensibilidade é contra-balanceado pelo número de exemplos incorretamente cobertos — 11 na regra inicial e 40 na regra final.

IF	**involved_nodes** ≤ 2.5	bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
	AND **irradiation** = no	68,	11,	29,	30,	138
THEN	**CLASS** = *no_recurrence*	[0.4928,	0.0797,	0.2101,	0.2174,	138]

Regra 6.10: Exemplo de regra inicial – breast – Sensibilidade

IF	**degree_of_malig** ≥ 1					
	AND **degree_of_malig** ≤ 3	bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
	AND **tumor_size** $\geq$ no	98,	40,	0,	0,	138
THEN	**CLASS** = *no_recurrence*	[0.7101,	0.2899,	0.0000,	0.0000,	138]

Regra 6.11: Exemplo de regra final – breast – Sensibilidade

Em geral, pode ser observado que os resultados obtidos pelo algoritmo evolutivo nos diferentes cenários utilizando a mesma população inicial e uma única medida como função de avaliação (simples-objetivo), não apresentam diferenças significativas entre si. Como esperado, dependendo da característica da medida considerada, uma regra com máximo valor dessa medida é construída, mas não necessariamente levando em conta o número de exemplos incorretamente cobertos pela regra. Utilizando funções multi-objetivo esse aspecto pode também ser considerado.

6.7.2 Funções de Avaliação Multi-Objetivo

Neste seção são analisados os resultados experimentais do Grupo 1 utilizando as funções de avaliação multi-objetivo *Fnc1* e *Fnc2*.

6.7.2.1 Fnc1 — *Novidade, Laplace e Suporte*

Na Tabela A.6 na página 145 são mostrados os resultados obtidos utilizando a função de avaliação multi-objetivo *Fnc1*. Para facilitar a análise, para cada conjunto de dados, esses resultados são mostrados graficamente nas Figuras B.8 à B.14 na página 160. Nessas figuras é mostrado, para cada cenário, os valores médios das medidas que participam da função de avaliação nas regras inicial e final.

Exceto para os conjuntos de dados cmc, heart e vehicle, ilustrados respectivamente nas Figuras B.9, B.10 e B.13, para os outros conjuntos de dados, além da somas das medidas acumuladas ser melhor, a função multi-objetivo *Fnc1* conseguiu uma melhor distribuição dos valores das medidas que dela participam. Isso é obtido diminuindo o valor da medida de Laplace, que em todos os casos domina as outras medidas[3], e incrementando os valores das medidas de novidade e suporte. Observe que o valor máximo da novidade de uma regra é 0.25.

Para o conjunto de dados cmc, os valores iniciais das medidas de novidade e suporte são muito pequenos, mas o algoritmo evolutivo consegue distribuir melhor os valores dessas medidas, em detrimento da medida de Laplace, diminuindo as diferenças entre elas — Figura B.9 na página 158. O mesmo é observado no conjunto de dados vehicle, ainda que neste caso os valores iniciais das medidas de novidade e suporte não são tão baixos como no caso do conjunto de dados cmc — Figura B.13 na página 160. Já no caso do conjunto de dados heart, pode ser observado que o valor da medida de Laplace diminui um pouco e o valor do suporte melhora, mas sem melhora significativa para a novidade. Neste caso há pouca variação entre os valores iniciais e finais dessas medidas na função multi-objetivo. Também pode ser observado que o comportamento do algoritmo evolutivo é semelhante em todos os cenários, exceto para o conjunto de dados cmc, que apresenta alguma variabilidade nos diferentes cenários.

Para ilustrar, a Regra 6.12 é a melhor regra inicial de um dos *folds* do conjunto de dados cmc e a Regra 6.13 a final. A Regra 6.12 apresenta os seguintes

[3]Observe que isso é esperado já que o conjunto de regras iniciais, como mencionado anteriormente, consiste de 80% de regras de conhecimento provenientes dos algoritmos de aprendizado $\mathcal{C}4.5$ e $\mathcal{CN}2$, os quais tentam, entre outros objetivos, maximizar essa medida.

valores: novidade=0.0067, Laplace=0.5557 e suporte=0.0258, enquanto que a Regra 6.13 apresenta: novidade=0.0564, Laplace=0.4618 e suporte=0.2283. Como pode ser observado, a Regra 6.13 final com um melhor equilíbrio entre os valores dessas medidas, é uma regra que prediz uma classe diferente que a regra inicial.

IF	**hocu** = 3					
	AND **nchi** $>$ 0					
	AND **nchi** $\leq$ 2					
	AND **wage** $\leq$ 37					
	AND **wedu** = 2	bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
	AND **work** = 1	4,	3,	76,	55,	736
THEN	**CLASS** = *1*	[0.0258,	0.0190,	0.5543,	0.4008,	736]

Regra 6.12: Exemplo de regra inicial – cmc – Multi-Objetivo *Fnc1*

IF	**nchi** $>$ 1.4					
	AND **wage** $<$ 39.08	bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
	AND **wrel** = 1	32,	50,	54,	16,	736
THEN	**CLASS** = *3*	[0.2283,	0.2649,	0.3886,	0.1182,	736]

Regra 6.13: Exemplo de regra final – cmc – Multi-Objetivo *Fnc1*

6.7.2.2 Fnc2 — *Sensibilidade e Especificidade*

No caso da função multi-objetivo *Fnc2*, que utiliza as medidas de sensibilidade e especificidade, utilizamos o espaço ROC (do inglês *Receiver Operating Characteristic*) (Fawcett, 2006) para visualizar e analisar os resultados. Esse espaço pode ser utilizado para a visualização e comparação de qualquer modelo de classificação binário, incluindo classificadores (que podem ou não serem compostos por um conjunto de regras) ou regras individuais.

No espaço ROC, o eixo das abscissas representa a taxa de falsos positivos (FP) e o eixo das ordenadas representa a taxa de verdadeiros positivos (TP), sendo que no caso de regras individuais, a regra R_i é representada pelo ponto $R_i = (FP_i, TP_i)$ nesse espaço. Note que a taxa de falsos positivos é equivalente a 1-especificidade da regra e a taxa de verdadeiros positivos é equivalente à sensibilidade, *i.e.* $FP = 1 - \frac{f_{\overline{bh}}}{\overline{h}}$ e $TP = \frac{bh}{h}$.

Alguns pontos no espaço ROC merecem destaque — Figura 6.1. O canto inferior esquerdo (0,0) representa a estratégia de classificar um exemplo como positivo (regra que sempre prediz negativo). O canto superior direito (1,1) representa a estratégia inversa de sempre classificar um exemplo como positivo (regra que sempre prediz positivo). O canto superior esquerdo (0,1) representa a classificação perfeita: todos os exemplos positivos são corretamente classificados como positivos e nenhum exemplo negativo é classificado como positivo.

Pontos que estão na diagonal principal representa uma estratégia de classificação aleatória: o ponto (0.2,0.2), por exemplo, representa a estratégia de classificar um exemplo como positivo 20% das vezes, e negativo no restante dos casos. Qualquer regra que caia nessa diagonal tem um desempenho igual ao aleatório. Pontos que estão abaixo dessa diagonal representam estratégias piores do que a aleatória. Dessa maneira, a região de maior interesse é representada pelo triângulo acima da diagonal principal. Um ponto no espaço

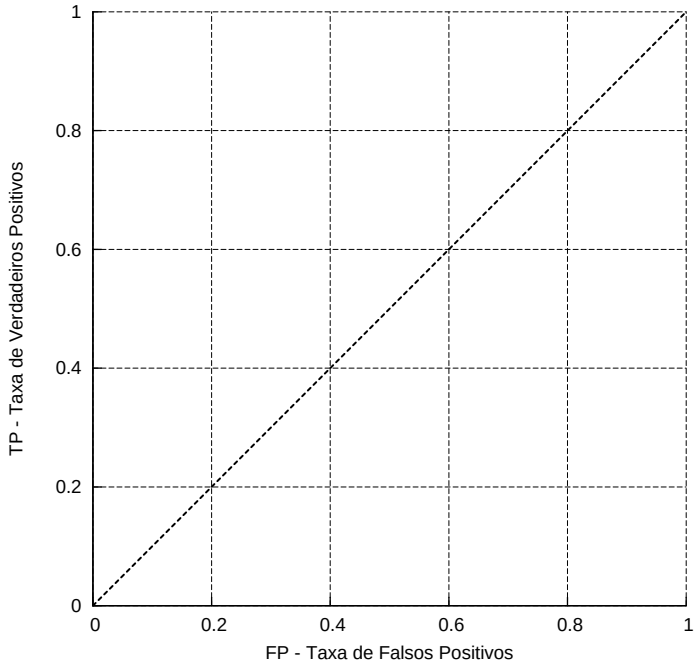

Figura 6.1: Espaço ROC

ROC é estritamente melhor que outro se ele estiver a noroeste (TP é maior e FP é menor) do que o primeiro. Entretanto, pontos a nordeste (TP é maior e FP também é maior) também podem ser interessantes, uma vez que representam regras com maior generalidade (que cobrem mais exemplos), pagando um preço de cobrir mais exemplos negativos. Dessa maneira, os pontos que ficam na região do "envelope" mais externo (*convex hull* ou diagonal superior), ou próximos a ele, são os de maior interesse (Provost & Fawcett, 2001).

Para o conjunto de dados breast, em geral não há uma melhoria ente a regra inicial e final. Como pode ser observado na Figura 6.2 , na maioria das vezes a taxa de verdadeiros positivos não incrementa (ou até diminui) com o incremento da taxa de falsos positivos. Somente para o cenário *Cnr2* isso não acontece.

Para os conjuntos de dados cmc e heart — Figuras 6.3 e 6.4 — houve pouca melhoria entre R_i e R_f. No caso de heart houve também cenários nos quais a

R_f foi pior. O que se observa é que em ambos conjuntos de dados o melhor resultado foi atingido com o cenário *Cnr2*, que é um cenário conservador por utilizar *crossover* com taxa de 80% e mutação com 5%.

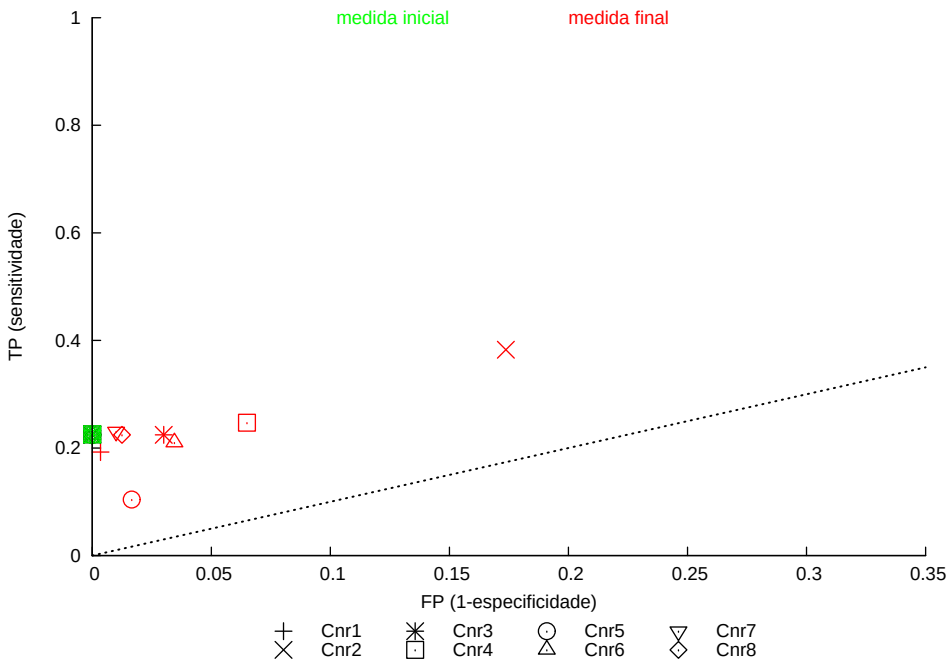

Figura 6.2: Grupo 1 – breast – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

Para o conjunto de dados nursery, independente do cenário, foi obtida a solução ótima do espaço ROC, ou seja, $TP = 1.0$ e $FP = 0$ — Figura 6.5 na página 106. Em todos os cenários, foi encontrada a regra "perfeita" que cobre a totalidade dos exemplos positivos e negativos corretamente. Essa regra foi também encontrada utilizando como medida simples-objetivo Laplace — Regra 6.3 na página 98.

Para o conjunto de dados spambase — Figura 6.6 na página 106 — pode ser observada uma melhoria em todos os cenários, a qual é menor no cenário *Cnr6*.

Para o conjunto de dados vehicle pode ser observado na Figura 6.7 que a maioria dos melhores resultados concentram-se nos cenários ímpares, *i.e.* aqueles nos quais a mutação é alta (40%).

Já para o conjunto de dados wbc — Figura 6.8 — pode ser observado que a melhoria entre a regra inicial e a final é mínima. Na realidade, para este conjunto de dados há poucas chances de melhoria, pois o TP da regra inicial esta próximo à 1.

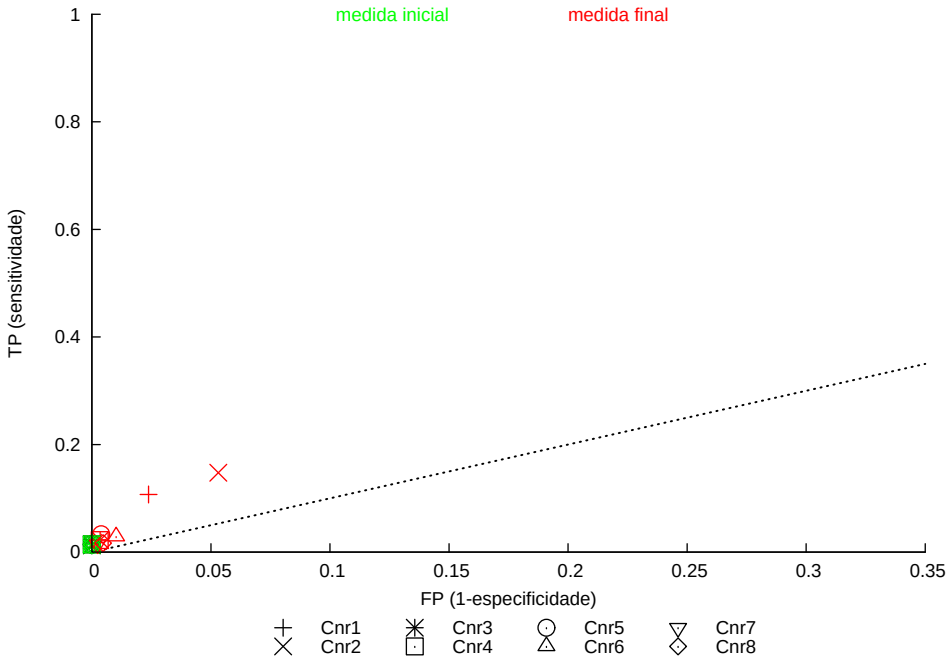

Figura 6.3: Grupo 1 – cmc – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

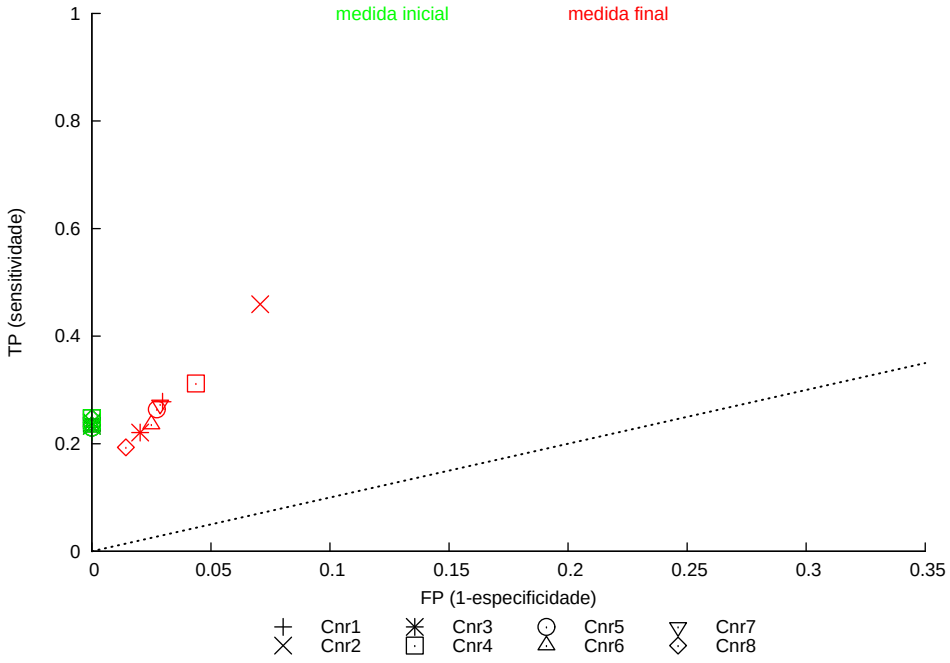

Figura 6.4: Grupo 1 – heart – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

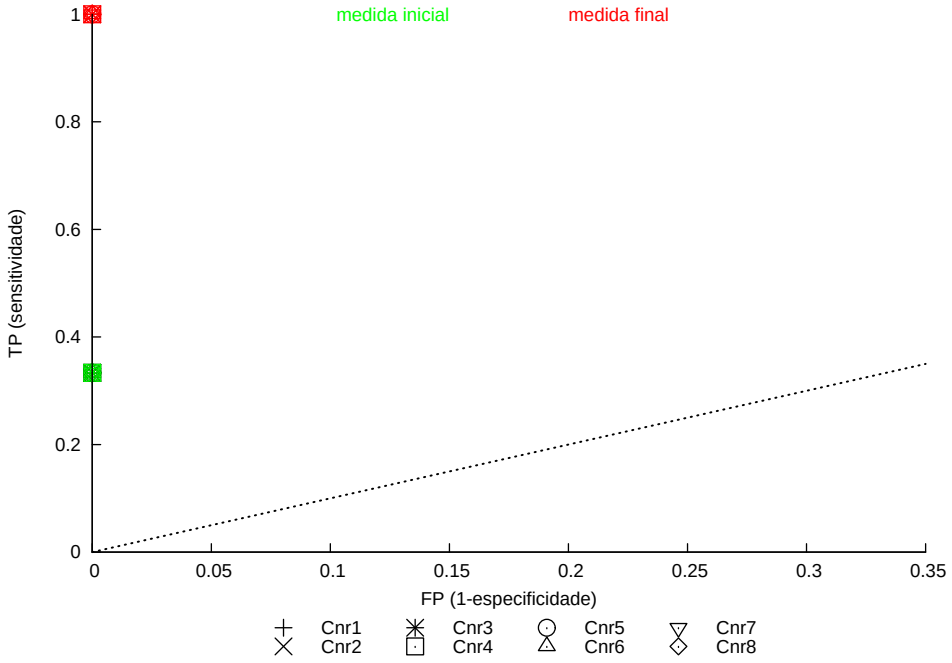

Figura 6.5: Grupo 1 – nursery – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

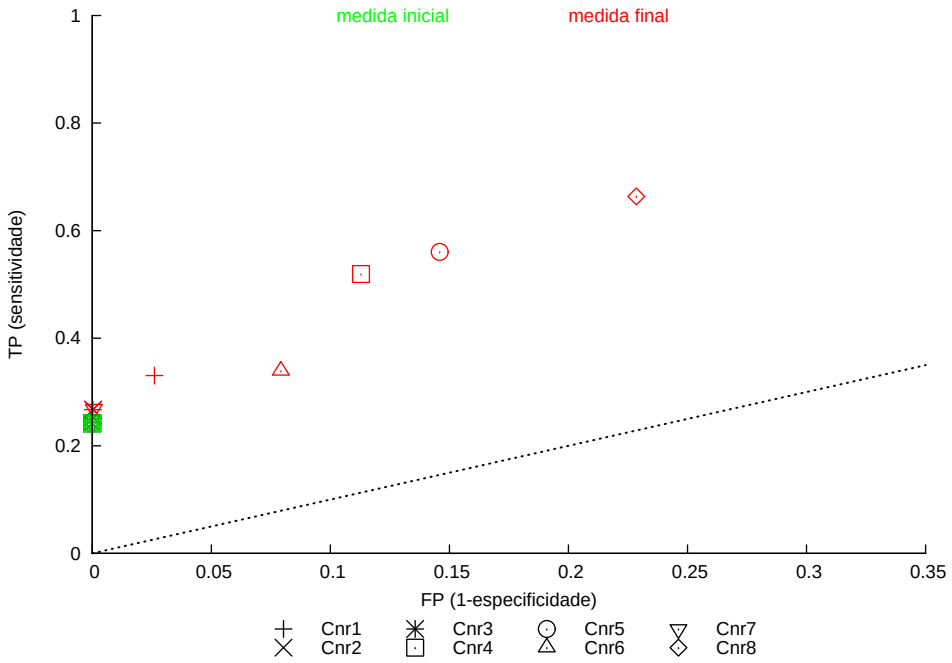

Figura 6.6: Grupo 1 – spambase – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

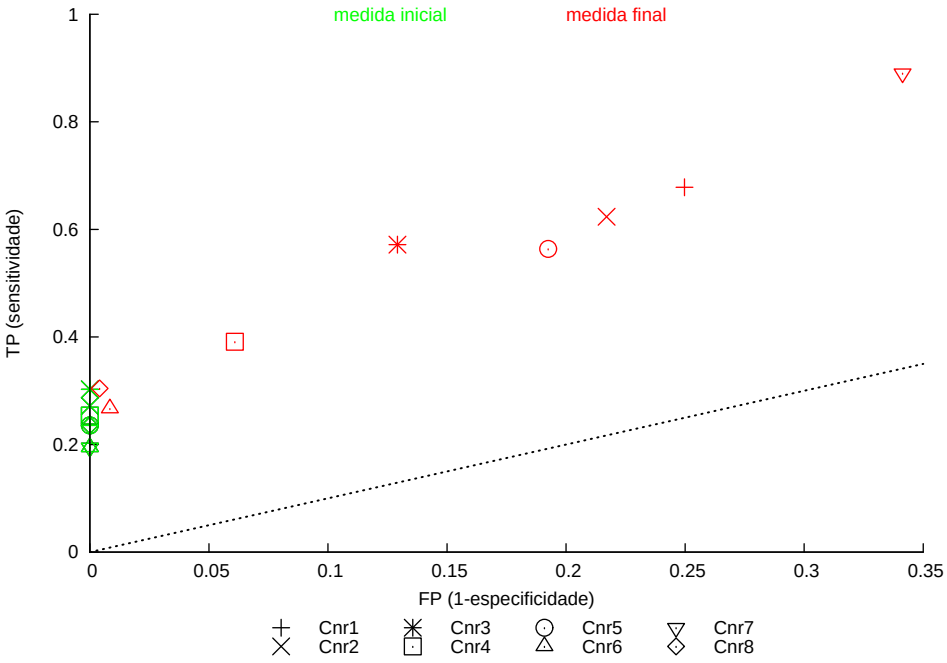

Figura 6.7: Grupo 1 – vehicle – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

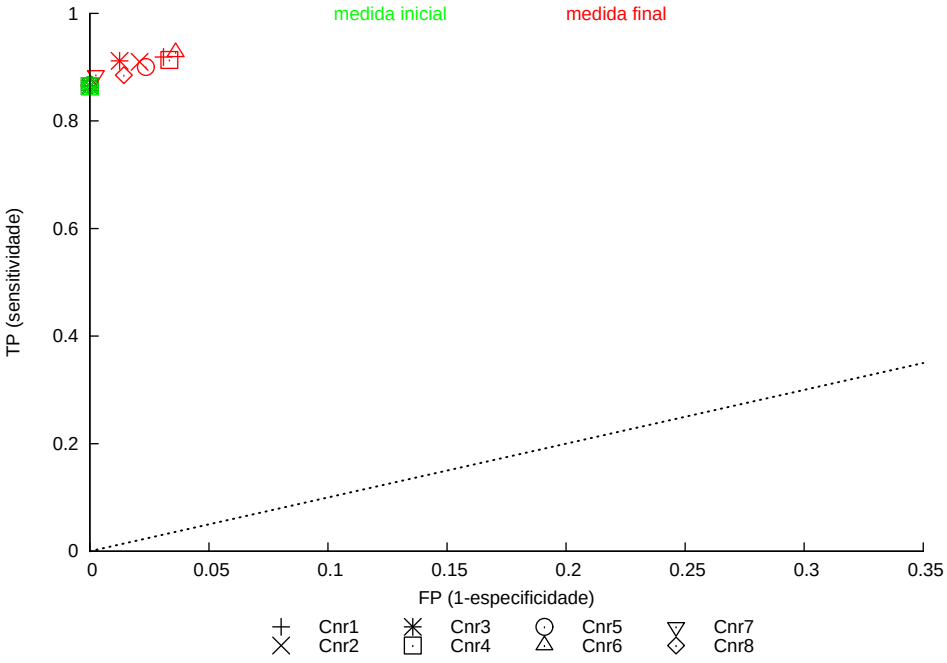

Figura 6.8: Grupo 1 – wbc – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

6.8 Resultados Experimentais — Grupo 2

Como mencionado, todos os experimentos do Grupo 2 foram realizados executando o algoritmo evolutivo utilizando diferentes populações iniciais com o objetivo de verificar se o algoritmo evolutivo consegue construir regras apropriadas utilizando as funções multi-objetivo independentemente do conjunto de indivíduos (regras) iniciais. Os resultados obtidos utilizando as duas funções multi-objetivo consideradas são descritas a seguir.

6.8.1 *Fnc1* — Novidade, Laplace e Suporte

Na Tabela A.8 na página 150 são mostrados os resultados obtidos utilizando esta função multi-objetivo. Também, semelhante à Seção 6.7.2.1, esses resultados são mostrados graficamente nas Figuras B.15 à B.21 na página 164. Analisando essas figuras é possível observar que o comportamento para cada conjunto de dados, sem considerar os valores das medidas que participam da função multi-objetivo, mas a variação das mesmas na regra inicial e final, é semelhante ao do Grupo 1, exceto nos seguintes casos:

- **heart** – no Grupo 1 a soma das medidas consideradas diminuiu um pouco na regra final na maioria dos cenários. Já no Grupo 2, essa soma é maior na regra final — Figuras B.10 e B.17.
- **breast** – além de apresentar uma maior variabilidade nos cenários quando comparada com os resultados do Grupo 1, no cenário *Cnr8*, a soma das medidas diminuiu — Figuras B.8 e B.15.

6.8.2 *Fnc2* — Sensibilidade e Especificidade

Para o conjunto de dados breast, ocorreram pequenos ganhos de cobertura no número de exemplos corretamente cobertos (TP), mas com um acréscimo no número de exemplos incorretamente cobertos (FP). Esse resultado pode ser confirmado pela curva ROC da Figura 6.9 .

Para o conjunto de dados nursery, independente do cenário, ao final foi obtida a solução ótima, ou seja, $TP = 1.0$ e $FP = 0$. Esse resultado pode ser confirmado pela curva ROC da Figura 6.10 . Analisando o gráfico é possível notar que as soluções iniciais variaram muito entre os cenários, dada a forma de inicializar a população inicial de regras, mas o melhor resultado é novamente alcançado.

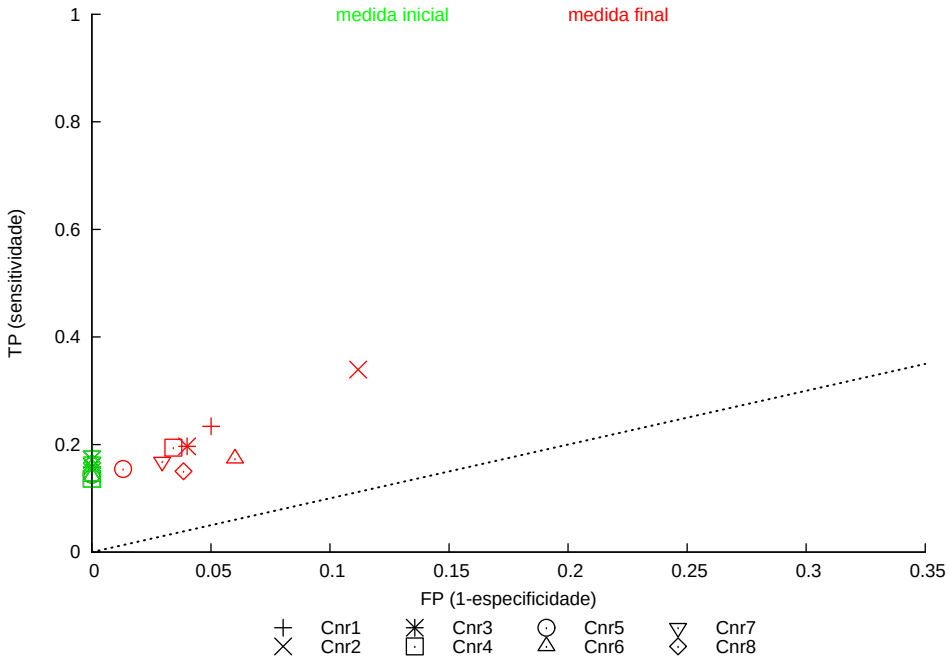

Figura 6.9: Grupo 2 – breast – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

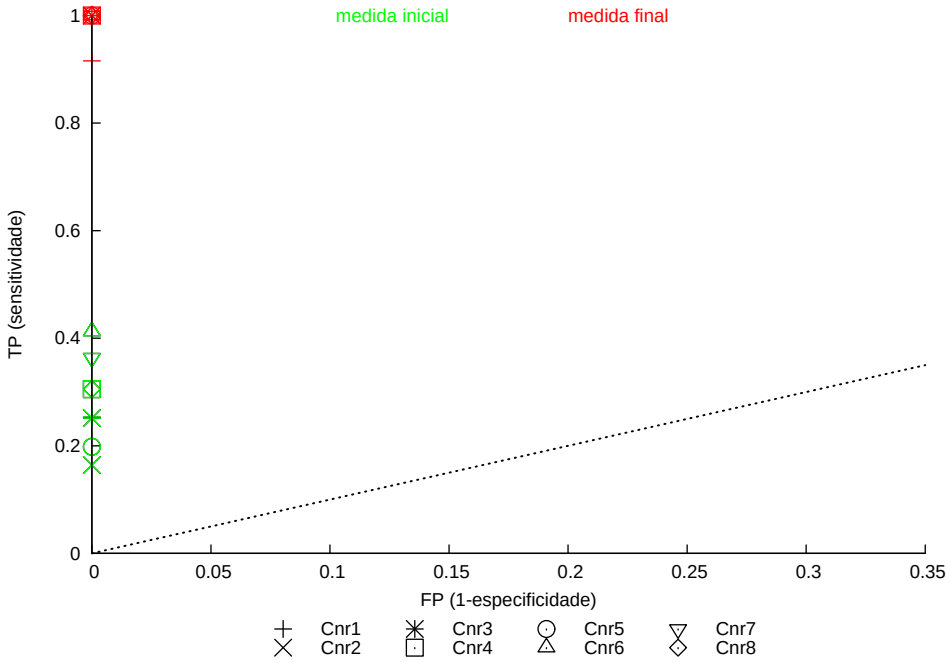

Figura 6.10: Grupo 2 – nursery – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

Para os conjuntos de dados spambase e vehicle, houve um aumento considerável no número de exemplos positivos corretamente cobertos, com destaque para o conjunto de dados vehicle, com um pequeno aumento no número de exemplos cobertos incorretamente, conforme curva ROC das Figuras 6.11 e 6.12, respectivamente.

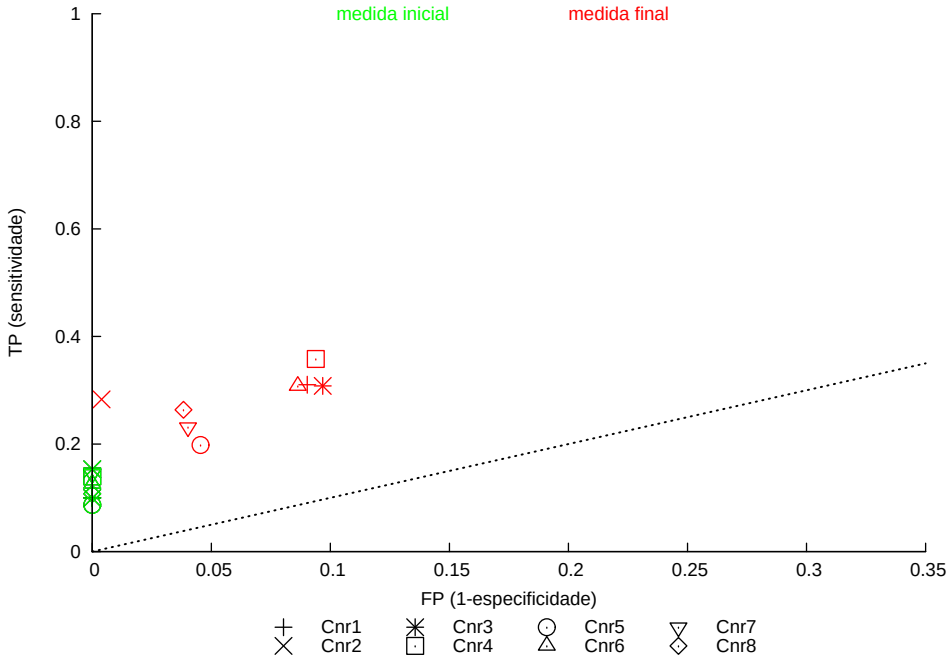

Figura 6.11: Grupo 2 – spambase – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multiobjetivo *Fnc2*

Para os conjuntos de dados cmc, heart e wbc — Figuras 6.13, 6.14 e 6.15 — as melhorias não foram muito significativas, embora tenham ocorrido. Nesses casos os melhores resultados foram alcançados com o cenário *Cnr2*, que é um cenário conservador por utilizar *crossover* com 80% e mutação com 5%.

Os resultados dos experimentos realizados utilizando as mesmas (Grupo 1) e diferentes (Grupo 2) regras iniciais do algoritmo evolutivo, mostram que em ambos casos, na grande maioria, o algoritmo evolui para uma solução final melhor com as características consideradas pela função de avaliação. Como esperado, a qualidade dos resultados depende fortemente do domínio $\mathcal{D}$ do conjunto de exemplos. Com relação aos diferentes cenários utilizados, os resultados variam dependendo do domínio. Em alguns casos, os resultados em alguns cenários são superiores, mas esses cenários variam. Para conjuntos de dados "bem comportados", tais como o conjunto de dados nursery, no qual os exemplos de classes diferentes estão bem separados, o algoritmo consegue

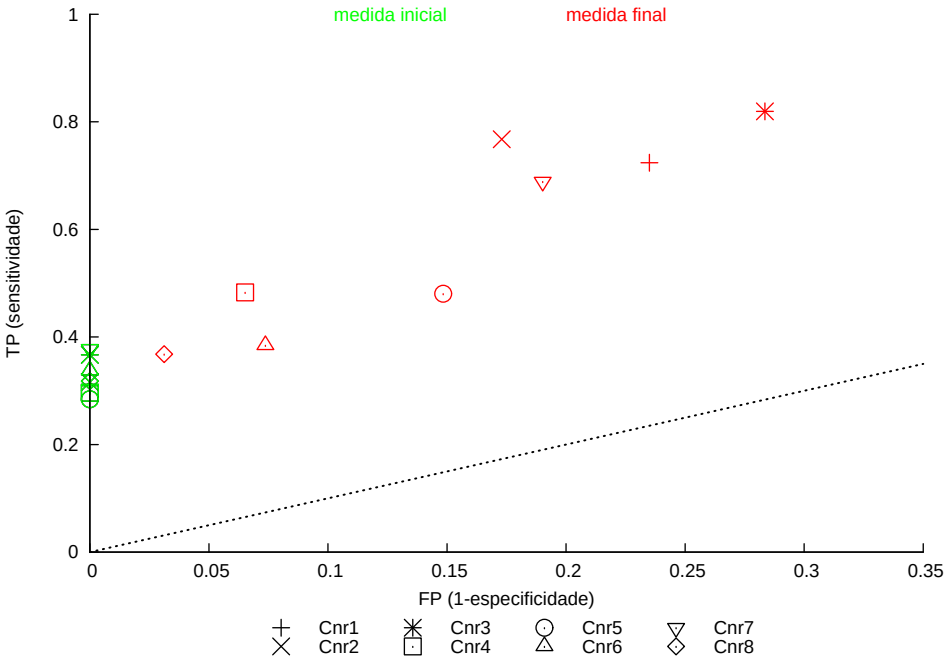

Figura 6.12: Grupo 2 – vehicle – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

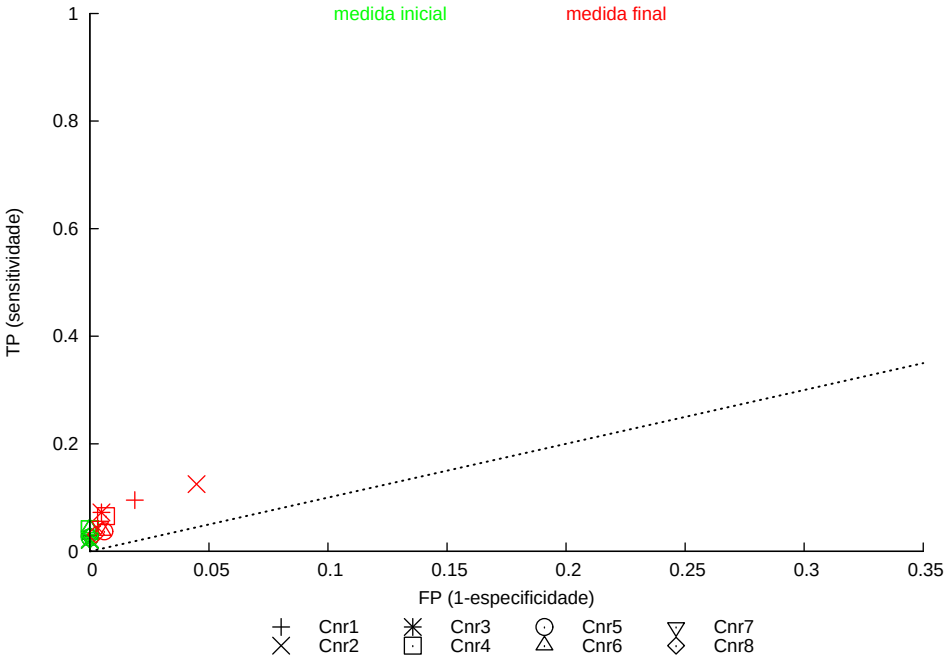

Figura 6.13: Grupo 2 – cmc – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

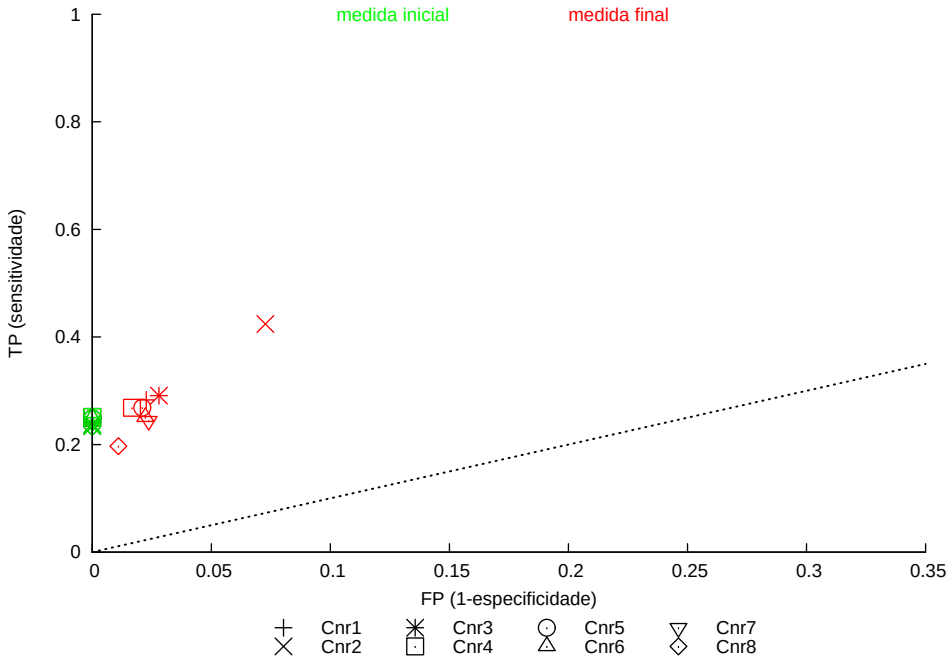

Figura 6.14: Grupo 2 – heart – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

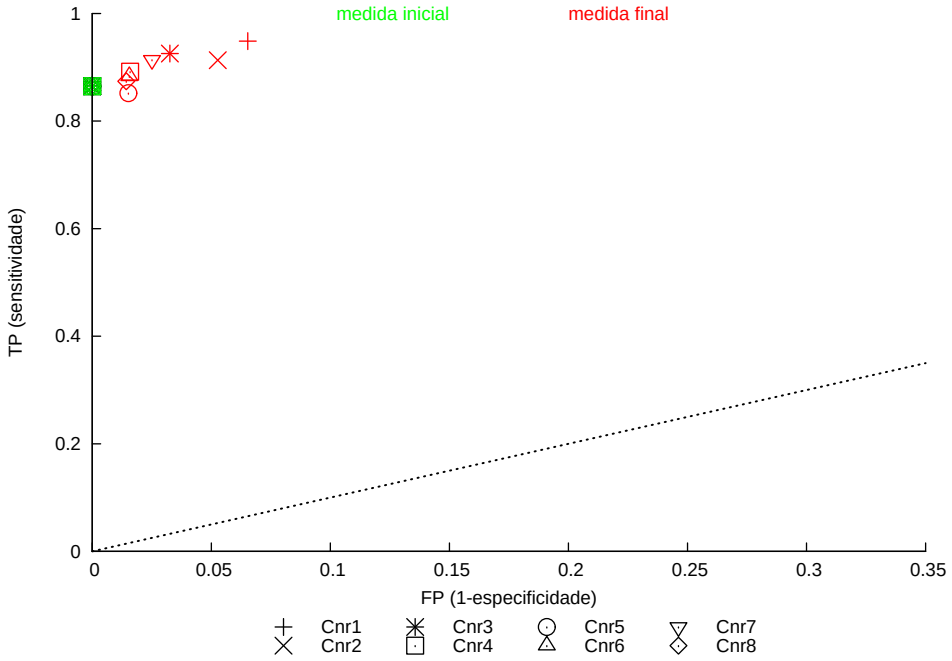

Figura 6.15: Grupo 2 – wbc – Gráfico ROC dos valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc2*

encontrar a melhor regra.

O fato do algoritmo convergir para boas soluções utilizando diferentes populações, indica que o algoritmo pode ser utilizado pelo especialista do domínio para também testar algumas hipóteses específicas sobre os dados. Isso pode ser realizado utilizando somente regras iniciais com condições presentes no conjunto de atributos que o especialista tem interesse em analisar. Se esses atributos não aparecem nas melhores soluções, isso pode ser considerado um indicativo da importância e da interação desses atributos na hipóteses levantada pelo especialista.

Consideramos que o campo de aplicação da nossa proposta é vasta e o ambiente computacional desenvolvido pode ser utilizado pelo especialista para realizar uma análise inteligente dos dados de um determinado domínio $\mathcal{D}$. Como mencionado, esta área de pesquisa necessita de avaliações experimentais para a validação, o que é praticamente impossível de ser realizado sem um ambiente apropriado para realizar experimentos, tais como o ambiente SNIFFERECLE proposto e implementado neste trabalho. Considere por exemplo os resultados experimentais apresentados neste capítulo. Para cada conjunto de dados foram considerados 8 cenários diferentes e foi realizado *5x2-cross-validation*, o que resulta em 80 execuções do algoritmo evolutivo. Além disso, foram realizados dois grupos de experimentos. No Grupo 1 o algoritmo foi executado utilizando como função de avaliação simples-objetivo cinco medidas de regras e duas funções de avaliação multi-objetivo, o que resulta em $80 \times 7 = 560$ execuções. No Grupo 2 o algoritmo foi executado utilizando duas funções de avaliação multi-objetivo, *i.e.* $80 \times 2 = 160$ execuções. Assim, para cada conjunto de dados, foram realizadas 720 execuções, ou seja, para os sete conjuntos de dados foi necessário um total de $7 \times 720 = 5040$ execuções do algoritmo evolutivo para obter os resultados apresentados neste capítulo, sendo 50% o número total de experimentos diferentes. Deve ser observado que além de ser possível organizar e disparar vários experimentos diferentes conjuntamente utilizando o SNIFFERECLE, é também de fundamental importância as facilidades que ele possui para, além de condensar os resultados obtidos nos experimentos, calculando médias de valores, desvio padrão e outros, fornecer também esses resultados em tabelas formatadas para o processador de textos LaTeX.

6.9 Tempo Aproximado de Execução do Algoritmo Evolutivo

Para fornecer uma idéia aproximada do tempo utilizado pelo algoritmo proposto para encontrar uma solução, foi utilizado como parâmetro o cenário *Cnr1*, a função de avaliação multi-objetivo *Fnc1* e todos os exemplos do conjunto de dados. Todos os tempos informados na Tabela 6.7 são tempos de processador obtidos após a convergência com 20 iterações. Os tempos foram mensurados em um *cluster* com 10 nós Dual Xeon 2.0 Ghz com 1 Gb de memória RAM. Vale ressaltar que na implementação do SNIFFERECLE — Seção5.8 — foi tomado o cuidado de lançar os processos em paralelo quando da chamada da $\mathcal{ECLE}$ para cada partição do *fold*. Assim, em um processo de *5x2-cross-validation*, com o equipamento que dispomos, os 10 processos podem ser executados em paralelo.

Conj. Dados	#Ex	#Atrib	Tempo (s)
breast	276	9	66
cmc	1473	9	294
heart	270	13	146
nursery	12960	8	5332
spambase	4601	57	4260
vehicle	846	18	790
wbc	683	10	306

Tabela 6.7: Tempo aproximado para uma execução do algoritmo evolutivo

Como pode ser observado, os tempos, para uma população de r indivíduos (regras), está relacionado com o número n de exemplos utilizados para calcular a matriz de contingência, cujos valores são utilizados pela função de avaliação. Para cada novo indivíduo criado, na próxima geração esse cálculo deve ser refeito. Para cada indivíduo, a complexidade do algoritmo que calcula essas informações é $O(n)$ (Prati et al., 2001). Assim, a complexidade para a população de r indivíduos é $O(r \times n)$. Como uma única geração não é suficiente, g gerações são criadas até a convergência. Portanto, a complexidade do algoritmo evolutivo para gerar uma solução é $O(r \times n \times g)$.

Em termos de uso efetivo de memória para armazenar a população de indivíduos iniciais do algoritmo evolutivo implementado neste trabalho — Figura 4.2 na página 47 — tem-se:

- *bit* de presença: 1 caracter, 1 *byte*;
- op: 2 caracteres, 2 *bytes*;

- vlr: valor real com ponto flutuante, 8 *bytes*;
- classe: *string*[4], 5 bytes;
- valores da matriz de contingência: valor real com ponto flutuante, 8 *bytes*;

Assim, para um conjunto de dados com M atributos o espaço utilizado para representar cada indivíduo na $\mathcal{ECLE}$ em bytes é: $((1\,byte + 2\,bytes + 8\,bytes) \times 2 \times M) + 5\,bytes + (8\,bytes \times 10)) = 22 \times M\,bytes + 85\,bytes$. Considerando uma população inicial composta por r regras, tem-se um total de alocação de memória de $(22 \times M\,bytes + 85\,bytes) \times r$ para a população inicial do algoritmo evolutivo.

6.10 Exemplo de Construção de Regras Utilizando Outra Abordagem

Como dito anteriormente, uma das abordagens para obter regras de conhecimento com propriedades específicas, é induzir um classificador simbólico e desse conjunto de regras extrair aquelas que contenham as propriedades desejadas pelo especialista (Freitas, 1998b).

Para exemplificar esse processo e comparar com a abordagem aqui proposta, foram induzidos classificadores com os indutores $\mathcal{CN}2$ e $\mathcal{C}4.5$ utilizando o conjunto de dados vehicle com todos os exemplos. Foram consideradas as medidas de Laplace, novidade e suporte como características a serem verificadas nas regras. Para o classificador induzido pelo $\mathcal{CN}2$, a melhor regra segundo esses critério é a Regra 6.14. Para o classificador induzido pelo $\mathcal{C}4.5$, a melhor

Laplace: 0.9620 Novidade: 0.0658 Suporte: 0.0887

		bh,	$b\overline{h}$,	$\overline{bh}$,	$\overline{b}h$,	n
IF	**MAX_LENGTH_ASPECT_RATIO** < 8					
	AND **PR_AXIS_ASPECT_RATIO** > 66	75,	0,	628,	143,	846
THEN	**CLASS** = *bus*	[0.0887,	0.0000,	0.7423,	0.1690,	846]

Regra 6.14: Melhor regra do $\mathcal{CN}2$ – vehicle – Multi-Objetivo *Fnc1*

regra segundo esse critério é a Regra 6.15.

Essas regras foram escolhidas desses classificadores utilizando o valor da função *Fnc1*.

[4]Em PERL a alocação de *strings* é feita de forma dinâmica.

Laplace: 0.9548 Novidade: 0.0944 Suporte: 0.1241

IF **ELONGATEDNESS** > 41
AND **HOLLOWS_RATIO** > 189
AND **MAX_LENGTH_ASPECT_RATIO** > 8 bh, $b\overline{h}$, $\overline{bh}$, $\overline{b}h$, n
AND **SKEWNESS_ABOUT_MAJOR_AXIS** > 63 105, 2, 645, 94, 846
THEN **CLASS** = *van* [0.1241, 0.0024, 0.7624, 0.1111, 846]

Regra 6.15: Melhor regra do $\mathcal{C}4.5$ – vehicle – Multi-Objetivo *Fnc1*

A regra construída pelo algoritmo evolutivo é a Regra 6.16. Ela foi construída utilizando a configuração do cenário *Cnr2*, que é a configuração mais freqüentemente utilizada como padrão para este tipo de algoritmo. Como previamente, foram utilizadas $N_{Regras} = 41$ regras iniciais, para compor a população inicial, das quais 17 foram construídas pelo $\mathcal{CN}2$, 17 pelo $\mathcal{C}4.5$ e 8 pelo $\mathcal{APRIORI}$ — Tabela 6.4 na página 91.

Laplace: 0.7708 Novidade: 0.1066 Suporte: 0.1523

IF **DISTANCE_CIRCULARITY** > 50.9
AND **ELONGATEDNESS** > 41
AND **ELONGATEDNESS** ≤ 61
AND **HOLLOWS_RATIO** > 189
AND **MAX_LENGTH_ASPECT_RATIO** > 7.3
AND **SCALED_VARIANCE_ALONG_MAJOR_AXIS** ≥ 132.1 bh, $b\overline{h}$, $\overline{bh}$, $\overline{b}h$, n
AND **SKEWNESS_ABOUT_MAJOR_AXIS** ≥ 59 129, 36, 611, 70, 846
THEN **CLASS** = *van* [0.1523, 0.0421, 0.7227, 0.0829, 846]

Regra 6.16: Melhor regra do algoritmo evolutivo – vehicle – Multi-Objetivo *Fnc1*

Na Tabela 6.8 são mostrados os valores de Laplace, novidade e suporte dessas regras.

Medida	$\mathcal{CN}2$-Regra 6.14	$\mathcal{C}4.5$-Regra 6.15	$\mathcal{ECLE}$-Regra 6.16
Laplace	0.9620	0.9548	0.7708
Novidade	0.0658	0.0944	0.1066
Suporte	0.0887	0.1241	0.1523

Tabela 6.8: Medidas das melhores regras obtidas

Como já mencionado, $\mathcal{CN}2$ e $\mathcal{C}4.5$, além de considerar a interação entre as regras, preocupam-se em maximizar o valor da medida de Laplace. Portanto, os valores da novidade e suporte das Regras 6.14 e 6.15 são os que resultam após maximização da medida de Laplace. No caso do algoritmo evolutivo ($\mathcal{ECLE}$), a combinação dos *rankings* das três medidas é considerado, conseguindo assim distribuir melhor os valores das medidas consideradas.

As Regras 6.15 do $\mathcal{C}4.5$ e 6.16 do algoritmo evolutivo predizem a mesma classe *van*. É interessante observar que as condições **ELONGATEDNESS** > 41 e

HOLLOWS_RATIO $>$ 189 aparecem em ambas regras, mas a primeira condição fica especializada pelo algoritmo evolutivo com a condição **ELONGATEDNESS** $\leq$ 61. Por outro lado, as condições **MAX_LENGTH_ASPECT_RATIO** $>$ 8 e **SKEWNESS_ABOUT_MAJOR_AXIS** $>$ 63 na Regra 6.15 foram generalizadas pelo algoritmo evolutivo para **MAX_LENGTH_ASPECT_RATIO** $>$ 7.3 e **SKEWNESS_ABOUT_MAJOR_AXIS** $\geq$ 59. Também, a regra construída pelo algoritmo evolutivo considera uma condição sobre um atributo não considerado pela Regra 6.15, a condição **SCALED_VARIANCE_ALONG_MAJOR_AXIS** $\geq$ 132.1

6.11 Considerações Finais

Neste capítulo foi apresentada a avaliação experimental da nossa proposta com o objetivo de comprovar sua adequabilidade na construção de regras de conhecimento com propriedades especificas. A avaliação foi realizada utilizando diversos conjuntos de dados com diferentes características. Para cada conjunto de dados o algoritmo evolutivo foi executado utilizando vários cenários (configurações), formas diferentes de construir a população inicial, bem como foram utilizadas diversas funções simples-objetivo e multi-objetivo, o que permite verificar o estabelecimento de um melhor equilíbrio entre os valores das medidas que participam das funções multi-objetivo.

Os resultados experimentais obtidos utilizando esses conjuntos de dados mostram a adequabilidade do algoritmo evolutivo proposto para construir regras de conhecimento com propriedades específicas, sem levar em consideração a interação entre regras. No próximo capítulo são apresentadas as conclusões deste trabalho.

Capítulo

7

CONCLUSÃO

O mal de quase todos nós é que preferimos sermos arruinados pelos elogios a sermos salvos pelas críticas.
Norman Vincent (1898–1993)

A proposta deste trabalho consiste na investigação de métodos do paradigma evolutivo para construir regras de conhecimento que apresentem propriedades desejáveis especificadas pelo usuário, sem levar em consideração a interação entre as regras. Dentre essas propriedades desejáveis, além daquelas usualmente utilizadas, como cobertura e precisão, espera-se que as regras construídas retratem conhecimento raro ou inesperado.

Para pesquisar essa possibilidade, foi proposto um algoritmo evolutivo que recebe como entrada um conjunto de regras *if-then* de conhecimento as quais podem ser construídas usando algoritmos de aprendizado simbólico, outros algoritmos ou podem ser manualmente construídas pelo próprio usuário. Essas regras iniciais são utilizadas para povoar a população inicial do algoritmo evolutivo. A rica estrutura por nós proposta para representar os indivíduos (regras) permite definir uma grande variedade de operadores evolutivos. Neste trabalho propomos três operadores de *crossover* que consideram a estrutura de representação do indivíduo de diferentes óticas: estrutural, em nível de

atributo e local. Ainda considerando a estrutura de representação, foram propostos dois tipos de mutação: estrutural e local. Para avaliar a aptidão dos indivíduos utilizando múltiplos objetivo, propomos a combinação de objetivos em uma função simples-objetivo utilizando *rankings*, a qual mostrou-se apropriada para considerar concomitantemente as propriedades especificadas pelo usuário na construção dos melhores indivíduos (regras).

Para avaliar a proposta deste trabalho foi implementada uma biblioteca de classes chamada $\mathcal{ECLE}$, que implementa o algoritmo evolutivo e faz parte do projeto DISCOVER, um projeto para aquisição e avaliação de conhecimento em constante desenvolvimento em nosso laboratório de pesquisa. Foi também implementado o ambiente SNIFFERECLE, o qual é uma extensão do ambiente SNIFFER do DISCOVER, para automatizar o processo de avaliação experimental do algoritmo evolutivo. Caso o processo de avaliação experimental não tivesse sido automatizado, consideramos que seria inviável realizar as diversas avaliações experimentais relacionadas com este trabalho. Para realizar a avaliação experimental do algoritmo evolutivo proposto, foram selecionados diversos conjuntos de dados utilizando como critério as suas características. Para cada conjunto de dados o algoritmo evolutivo foi executado utilizando vários cenários (configurações), formas diferentes de construir a população inicial, bem como funções multi-objetivo.

Foi comprovado que utilizando, o mesmo ou diferentes, conjuntos de regras iniciais para construir a população do algoritmo evolutivo, ele evolui na grande maioria das vezes para uma solução final melhor, que leva em conta as características consideradas pela função de avaliação. Como esperado, os resultados dependem fortemente do domínio do conjunto de dados. Para conjuntos de dados "bem comportados", tais como o conjunto de dados nursery, no qual os exemplos de classes diferentes estão bem separados, o algoritmo consegue sempre encontrar a melhor regra de uma dessas classes.

7.1 Principais Contribuições e Limitações

A construção de regras de conhecimento com propriedades específicas, que podem ser especificadas pelo especialista, utilizando algoritmos evolutivos, tem sido pouco explorada na literatura. Assim, consideramos que o algoritmo evolutivo proposto, bem como sua implementação na $\mathcal{ECLE}$ e o ambiente SNIFFERECLE são as principais contribuições deste trabalho.

Quanto às publicações, o projeto da biblioteca de classes, a qual implementa

o algoritmo evolutivo proposto, bem como facilidades para a execução de experimentos, foi publicado na forma de um relatório técnico publicado pelo ICMC-USP (Giusti et al., 2006). A descrição da $\mathcal{ECLE}$ e os resultados considerando uma medida de avaliação simples-objetivo foram publicados em uma conferência (Pila et al., 2006a). A proposta de utilizar *rankings* como forma de compor várias medidas de avaliação de regras em uma única função simples-objetivo foi publicada em um *workshop* vinculado à uma conferência internacional (Pila et al., 2006b). Resultados experimentais iniciais da proposta deste trabalho foram publicados em outra conferência internacional (Pila et al., 2006c). Atualmente, encontra-se em preparação um artigo contendo os resultados obtidos neste trabalho, a ser submetido ao periódico *AI-Communications* (Pila & Monard, 2007).

Quanto as limitações, ela está relacionada com a inicialização proposta nos indivíduos iniciais, na qual as condições que não aparecem na regra para atributos quantitativos. Deve ser observado que a inicialização dessas condições simplificam muito a implementação dos operadores. Entretanto, durante a execução do algoritmo, quando uma dessas condições que não aparecem na regra original representada por esse indivíduo, *i.e.* encontra-se desativada, passa a estar ativada, a sua contribuição é nula, pois os valores atualmente utilizados para a inicialização são, respectivamente, o mínimo e o máximo valores do atributo quantitativo que participa nessa condição. Assim, consideramos que a maneira de inicializar esse tipo de condição para atributos quantitativos pode ser melhor analisada.

Também, pode ser observado que o usuário pode escolher que os pais de novos indivíduos sejam da mesma classe, *i.e.* as regras que eles representam predizem a mesma classe. Nesse caso, na implementação realizada a probabilidade de ocorrência do *crossover* é aparente, decrementando a ocorrência real desse operador.

7.2 Trabalhos Futuros

Como o algoritmo evolutivo proposto possui uma estrutura de representação dos indivíduos bastante rica em informação, outros operadores de evolutivos podem ser propostos. Neste trabalho usamos uma forma de combinar valores numéricos por ocasião do *crossover* local. Neste sentido, outros métodos podem ser pesquisados e facilmente incorporados na $\mathcal{ECLE}$.

Quanto ao processo de inicialização de regras proposto para as condições com

atributos qualitativos que não aparecem no corpo da regra, consideramos que outros valores limitantes devem ser melhor pesquisados. Por exemplo, o limitante inferior poderia ser inicializado com o valor referente ao 1º quartil dos valores desse atributo, enquanto que o limitante superior poderia ser inicializado com o valor referente ao 3º quartil.

Quanto à função multi-objetivo, a qual é transformada em uma função multi-objetivo utilizando *rankings*, seria interessante comparar os resultados com os obtidos por métodos que procuram pelo conjunto de soluções na fronteira de Pareto. Também, quanto à solução proposta utilizando *rankings*, utilizamos a média aritmética ou harmônica para combinar os *ranks* em uma função de avaliação multi-objetivo. Porém, outros métodos de combinação merecem ser melhor investigados.

Outra facilidade a ser considerada, é utilizar o algoritmo evolutivo para encontrar a melhor regra com as características especificadas pelo usuário e, após, retirar todos os exemplos que são cobertos corretamente por essa regra e repetir o processo utilizando os exemplos restantes para encontrar a próxima melhor regra com as mesmas características.

Quanto aos métodos propostos neste trabalho para combinar os *rankings* na função de avaliação simples-objetivo (*i.e.*, votação (*rank*) ponderada usando a média aritmética ou harmônica), vários outros métodos de votação e combinação podem ser utilizados e serão investigados e implementados futuramente.

REFERÊNCIAS BIBLIOGRÁFICAS

Agarwal, S., Cortes, C., & Herbrich, R., editores (2005). *Proceedings of the NIPS'2005 Workshop on Learning to Rank*. Whistler, BC, Canada. Eletronic available at http://web.mit.edu/shivani/www/Ranking-NIPS-05/. Citado na página 63.

Agrawal, R. & Srikant, R. (1998). Fast algorithms for mining association rules. páginas 580–592, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. Citado na página 90.

Alpaydin, E. (2004). *Introduction to Machine Learning*. MIT Press. Citado na página 8.

Bagley, J. D. (1967). *The behavior of adaptatve systems which employ genetic and correlation algorithms*. Tese de Doutorado, Universidade de Michigan. Citado na página 29.

Baranauskas, J. A. & Batista, G. E. A. P. A. (2000). O projeto DISCOVER: Idéias iniciais. Comunicação pessoal. Citado na página 69.

Baranauskas, J. A. & Monard, M. C. (2000). Reviewing some machine learning concepts and methods. Relatório Técnico 102, ICMC-USP, São Carlos, SP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_102.ps.zip. Citado na página 12.

Batista, G. E., Prati, R. C., & Monard, M. C. (2004a). A study of the behavior of several methods for balancing machine learning data. *SIGKDD Explorations: Special issue on Learning from Imbalanced Datasets*, 6(1):20–29. Citado na página 88.

Batista, G. E. A. P. A. (2001). Sintaxe padrão do arquivo de exemplos do projeto DISCOVER. http://www.icmc.sc.usp.br/~gbatista/Discover/SintaxePadraoFinal.htm. Citado nas páginas 68, 70 e 77.

Batista, G. E. A. P. A. (2003). Pré-processamento de dados em aprendizado de máquina supervisionado. Tese de Doutorado, ICMC-USP, http://www.icmc.usp.br/~gbatista/pdfs/TeseDoutorado.pdf. Citado nas páginas 71, 83, 84 e 89.

Batista, G. E. A. P. A. & Monard, M. C. (2003a). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5):519–533. Citado na página 71.

Batista, G. E. A. P. A. & Monard, M. C. (2003b). Descrição da arquitetura e do projeto do ambiente computacional DISCOVER LEARNING ENVIRONMENT — DLE. Relatório Técnico 187, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_187.pdf. Citado nas páginas 70, 71, 73 e 84.

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004b). A study of the behavior of several methods for balancing machine learning training data. *SIGKDDv Explorations*, 6(1):20–29. Special issue on Learning from Imbalanced Datasets. http://www.acm.org/sigs/sigkdd/explorations/. Citado na página 71.

Bäck, T., Fogel, D. B., & Michalewicz, T. (2000). *Evolutionary Computation 1 - Basic Algorithms and Operators*. Institute of Phisics Publishing - IoP. Citado nas páginas 2, 26, 41, 124, 125, 126, 127, 128, 129, 130 e 131.

Bernardini, F. C. (2006). *Combinação de Classificadores Utilizando Medidas de Regras de Conhecimento e Algoritmos Genéticos*. Tese de Doutorado, ICMC/USP, São Carlos, SP. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-29092006-110806/. Citado na página 18.

Blickle, T. (2000). *Tournament selection*, capítulo 24, páginas 181–186. Volume I de Bäck et al. (2000). Citado na página 36.

Booch, G., Jacobson, I., Rumbaugh, J., & Rumbaugh, J. (1998). *The Unified Modeling Language User Guide*. Addison-Wesley. Citado na página 73.

Bosworth, J., Foo, N., & Zeigler, B. P. (1972). Comparison of genetic algorithms with conjugate gradient methods. *National Aeronautics and Space Administration, Washington, DC*. Citado na página 29.

Bremermann, J. H. (1962). Optimization through evolution and recombination. *Self-Organizing Systems, M. C. Yovits et al.* Citado na página 26.

Bremermann, J. H., Rogson, M., & Salaff, S. (1965). Search by evolution. Em m. Maxfield, Callahan, A., & Fogel, L. J., editores, *Proceedings of 2nd.*

Cybernetic Science Symposium, páginas 157–167, Whasington, DC. Citado na página 26.

Carvalho, A. C. P. L. F., Braga, A. P., & Ludermir, T. B. (2003). *Computação Evolutiva*, capítulo 9, páginas 225–248. Volume 1 de Rezende (2003), 1 edição. Citado na página 38.

Cavicchio, D. J. (1970). *Adaptative search using simulated evolution*. Tese de Doutorado, Universidade de Michigan. Citado na página 29.

Chiara, R. (2003). Aplicações de técnicas de *data mining* em *logs* de servidores web. Dissertação de Mestrado, ICMC-USP. Citado na página 72.

Clark, P. & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. Em Kodratoff, Y., editor, *Proc. of the 5th European Conference EWSL 91*, páginas 151–163. Springer-Verlag. Citado na página 73.

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Mach. Learn.*, 3(4):261–283. Citado na página 90.

Coello, C. A. C. (2006). Evolutionary multi-objective optimization: A historial view of the field. *IEEE Comput. Int. Mag.*, páginas 28–36. Citado nas páginas 40 e 59.

Coello, C. A. C., Veldhuizen, D. A. V., & Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems*. Kluwer Academic, New York, NY, USA. Citado nas páginas 3, 40, 41 e 59.

Darwin, C. R. (1859). *On the Origin of Species*. John Murray, London. Citado nas páginas 23 e 25.

De Jong, K., Fogel, D. B., & Schwefel, H.-P. (2000). *A history of evolutionary computation*, capítulo 6, páginas 40–58. Volume I de Bäck et al. (2000). Citado na página 26.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptative system*. Tese de Doutorado, Universidade de Michigan. Citado na página 29.

Deb, K. (2000). *Introduction to selection*, capítulo 22, páginas 166–171. Volume I de Bäck et al. (2000). Citado na página 35.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, New York, NY, USA. Citado nas páginas 3 e 41.

Dietterich, T. G. (1998). Approximate statistical test for comparing supervised classification learning algorithms. *Neural Comp.*, 10(7):1895–1923. Citado na página 94.

Eshelman, L. J. (2000). *Genetic Algorithms*, capítulo 8, páginas 64–80. Volume I de Bäck et al. (2000). Citado na página 30.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874. Citado na página 102.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54. Citado na página 18.

Fayyad, U. M., Piatetsky-Shapiro, G., & Smyth, P. (1996b). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34. Citado na página 18.

Fisher, R. A. (1958). *The genetic theory of natural selection*. New York: Dover. Citado na página 26.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). On the evolution of artificial intelligence. Em *Artificial Intelligence through Simulated Evolution*, páginas 131–156, New York, Wesley. Citado na página 26.

Freitas, A. A. (1998a). A multi-criteria approach for the evaluation of rule interestingness. Em *Proc.of the International Conference on Data Mining*, páginas 7–20, Rio de Janeiro, RJ. Citado na página 17.

Freitas, A. A. (1998b). On objective measures of rule surprisingness. Em *Principles of Data Mining & Knowledge Discovery: Proc.of the Second European Symp. Lecture Notes in Artificial Intelligence*, volume 1510, páginas 1–9. Citado nas páginas 2, 3, 17 e 115.

Freitas, A. A. (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag. Citado nas páginas 33, 37, 40, 41, 47 e 50.

Freitas, A. A. (2004). A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Explor. Newsl.*, 6(2):77–86. Citado na página 59.

Friedberg, R. M. (1958). A learning machine: part I. *IBM J.*, 2:2–13. Citado na página 26.

Friedberg, R. M., Dunham, B., & North, J. H. (1959). A learning machine: part II. *IBM J.*, 3:282–287. Citado na página 26.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley. Citado na página 71.

Ghosh, A. & Nath, B. (2004). Multi-objective rule mining using genetic algorithms. *Inf. Sci.*, 163(1-3):123–133. Citado na página 3.

Giusti, R., Pila, A. D., & Monard, M. C. (2006). Arquitetura, projeto e implementação da biblioteca Evolutionary Computing Learning Envirionment ($\mathcal{ECLE}$) para construir regras de conhecimento com propriedades específicas. Relatório Técnico 289, ICMC-USP, São Carlos. `ftp://ftp.icmc.usp.br/pub/BIBLIOTECA/transf/RT_289.pdf`. Citado nas páginas 5, 74, 80 e 121.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optmization, and Machine Learning*. Addison-Wesley Publishing Company. Citado nas páginas 20, 26, 29, 32 e 50.

Gould, S. J. (2002). *The Structure of Evolutionary Theory*. Belknap Press of Harvard University Press, Cambridge, USA. Citado na página 24.

Grefenstette, J. (2000a). *Proportional selection and sampling algorithms*, capítulo 23, páginas 172–180. Volume I de Bäck et al. (2000). Citado na página 36.

Grefenstette, J. (2000b). *Rank-based selection*, capítulo 25, páginas 187–194. Volume I de Bäck et al. (2000). Citado na página 37.

Han, J. & Kamber, M. (2006). *Data mining : concepts and techniques*. Morgan Kaufmann, Boston, Elsevier. Citado na página 1.

Haupt, R. L. & Haupt, S. E. (1998). *Practical Genetic Algorithms*. Wiley-interscience Publication. Citado nas páginas 20, 26, 29, 35 e 36.

Heitkoetter, J. & Beasley, D. (2001). The hitch-hiker's guide to evolutionary computation: A list of frequently asked questions (FAQ). Disponível via FTP anônimo em `ftp://rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic/` ou USENET: `http://groups.google.com/groups?group=comp.ai.genetic`. Citado na página 26.

Herrera, F., Lozano, M., Pérez, E., Sanchez, A. M., & Villar, P. (2002). Multiple crossover per couple with selection of the two best offspring: An experimental study with the blx-alpha crossover operator for real-coded genetic algorithms. Em *IBERAMIA 2002: Proceedings of the 8th Ibero-American Conference on AI*, páginas 392–401, London, UK. Springer-Verlag. Citado na página 55.

Holland, J. H. (1967). Nonlinear environments permitting efficient adaptation. *Computer and Information Siences II, New York: Academic*. Citado na página 26.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press. Second Edition, MIT Press, 1992. Citado na página 29.

Hollstein, R. B. (1971). *Artificial genetic adaptation in computer control systems*. Tese de Doutorado, Universidade de Michigan. Citado na página 29.

Huang, J. & Ling, C. X. (2006). Constructing ensembles for better ranking. Em *Sixt IEEE International Conference on Data Mining (ICDM'06)*, páginas 902–906, Hong Kong. IEEE Computer Society. Citado na página 63.

Ishibuchi, H., Nakashima, T., & Murata, T. (2001). Three objective genetics-based machine learning for linguisitc rule extraction. *Information Science*, 136(1-4):109–133. Citado na página 3.

Ishibuchi, H. & Yamamoto, T. (2004). Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems*, 141:59–88. Citado na página 3.

Kendall, M. G. (1990). *Rank correlation methods*. A Charles Griffin Book. Citado na página 62.

Kinnear Jr., K. E. (2000). *Derivative methods in genetic programming*, capítulo 11, páginas 103–113. Volume I de Bäck et al. (2000). Citado na página 30.

Koza, J. R. (1992). *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. MIT Press. Citado na página 22.

Lavrač, N., Flach, P., & Zupan, B. (1999). Rule evaluatin measures: A unifying view. *Lecture Notes in Artificial Inteligence*, (1634):174–185. Citado nas páginas 4, 13, 16 e 17.

Metz, J. (2006). Interpretação de clusters gerados por algoritmos de clustering hierárquico. Dissertação de Mestrado, ICMC-USP, `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-14092006-090701/`. Citado na página 71.

Michalewicz, Z. (1997). *Genetic Algorithms + Data Structures = Evolution Programs*. IE-Springer-Verlag. Citado nas páginas 26, 30, 33, 39 e 46.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. The MIT Press. Citado nas páginas 22, 26 e 32.

Mitchell, M. & Taylor, C. E. (1999). Evolutionary computation: An overview. *Annual Review of Ecology and Systematics*, 20:593–616. http://citeseer.nj.nec.com/mitchell99evolutionary.html. Citado na página 31.

Monard, M. C. & Baranauskas, J. A. (2003). *Conceitos sobre Aprendizado de Máquina*, capítulo 4, páginas 89–114. Volume 1 de Rezende (2003), 1 edição. Citado na página 10.

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. Citado na página 88.

Parpinelli, R., Lopes, H., & Freitas, A. (2002). Data mining with an ant colony optimization algorithm. *IEEE Trans on Evolutionary Computation, special issue on Ant Colony Algorithms*, 6(4). http://www.cs.ukc.ac.uk/pubs/2002/1403. Citado na página 22.

Pila, A. D., Giusti, R., & Monard, M. C. (2006a). Um sistema evolutivo para a construção de regras de conhecimento com propriedades específicas. Em *Anais da 32a Conferência Latinoamericana de Informática*, páginas 10–12, CLEI, Santiago, Chile. CLEI. Citado nas páginas 5 e 121.

Pila, A. D., Giusti, R., & Monard, M. C. (2006b). Uma proposta para criar regras de conhecimento com propriedades específicas. Em Rezende, S. O. & da Silva Filho, A. C. R., editores, *1st Workshop on Computational Intelligence (WCI'06) in the Proceedings of International Joint Conference, 10th Ibero-American Artificial Intelligence Conference, 18th Brazilian Artificial Intelligence Symposium, 9th Brazilian Neural Networks Symposium, IBERAMIA-SBIA-SBRN*. ICMC-USP. Citado nas páginas 5 e 121.

Pila, A. D., Giusti, R., Prati, R. C., & Monard, M. C. (2006c). A multi-objective evolutionary algorithm to build knowledge classification rules with specific properties. Em *HIS '06: Proceedings of the Sixth International Conference on Hybrid Intelligent Systems*, página 41, Washington, DC, USA. IEEE Computer Society. http://dx.doi.org/10.1109/HIS.2006.6. Citado nas páginas 5 e 121.

Pila, A. D. & Monard, M. C. (2007). Building isolated knowledge rules with specific proporties using an evolutionary algorithm. *AI Communications*. (a ser submetido). Citado na página 121.

Porto, V. W. (2000). *A history of evolutionary computation*, capítulo 10, páginas 89–102. Volume I de Bäck et al. (2000). Citado nas páginas 27 e 28.

Prati, R. C. (2003). O *framework* de integração do sistema DISCOVER. Dissertação de Mestrado, ICMC-USP. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-20082003-152116/. Citado na página 70.

Prati, R. C. (2006). *Novas abordagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos*. Tese de Doutorado, ICMC/USP São Carlos. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-01092006-155445/. Citado na página 63.

Prati, R. C., Baranauskas, J. A., & Monard, M. C. (2001). Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Relatório Técnico 137, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip. Citado nas páginas 44, 45, 70, 71 e 114.

Prati, R. C., Batista, G. E. A. P. A., & Monard, M. C. (2004). Class imbalances *versus* class overlapping: an analysis of a learning system behavior. Em *Mexican International Conference on Artificial Intelligence, LNAI 2972*, páginas 312–321. Springer-Verlag. Citado na página 71.

Provost, F. & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231. Citado na página 103.

Quinlan, J. R. (1988). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, California. Citado nas páginas 73 e 90.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. *Royal Aircraft Eatabilishment Library Translation 1122, Farnborough, Hants, UK*. Citado nas páginas 26 e 28.

Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações*. Editora Manole, Barueri, SP, Brasil. Citado nas páginas 125, 129 e 130.

Rezende, S. O., Pugliesi, J. B., Melanda, E. A., & Paula, M. F. (2003). *Mineração de Dados*, capítulo 12, páginas 307–336. Volume 1 de Rezende (2003), 1 edição. ISBN 85-204-1683-7. Citado na página 18.

Romao, W., Freitas, A., & Gimenes, I. (2004). Discovering interesting knowledge from a science & technology database with a genetic algorithm. *Applied Soft Computing*, 4(2):121–137. Citado na página 3.

Rosenberg, R. S. (1967). *Simulation of genetic population with biochemical properties*. Tese de Doutorado, Universidade de Michigan. Citado na página 29.

Rudolph, G. (2000). *Evolution strategies*, capítulo 9, páginas 81–88. Volume I de Bäck et al. (2000). Citado nas páginas 28 e 29.

Russell, S. & Norvig, P. (2003). *Inteligência Artificial*. Campus Editora, São Paulo. Citado na página 10.

Schwefel, H.-P. & Rudolph, G. (1995). Contemporary evolution strategies. Em *European Conference on Artificial Life*, páginas 893–907. `http://citeseer.nj.nec.com/schwefel95contemporary.html`. Citado na página 29.

Setzkorn, C. & Paton, R. (2003). Merbis - a multi-objective evolutionary rule base induction system. Citado na página 3.

Smaldon, J. & Freitas, A. (2006). A new version of the ant-miner algorithm discovering unordered rule sets. Em et al., M. K., editor, *Proc. Genetic and Evolutionary Computation Conference (GECCO-2006)*, páginas 43–50. ACM Press. Citado na página 3.

Smith, R. E. (2000). *Learning classifier systems*, capítulo 12, páginas 114–123. Volume I de Bäck et al. (2000). Citado nas páginas 31 e 33.

Voltolini, R. F. (2006). Discretização e geração de gráficos de dados em aprendizado de máquina. Dissertação de Mestrado, ICMC-USP, `http://www.teses.usp.br/teses/disponiveis/55/55134/tde-24012007-144841/`. Citado nas páginas 71 e 89.

Wall, L., Christiansen, T., & Schwartz, R. L. (1996). *Programming Perl*. O'Reilly & Associates, 2 edição. Citado na página 70.

Apêndice

A

Resultados Experimentais — Tabelas

Neste apêndice são somente apresentadas as tabelas referentes aos resultados experimentais dos Grupos 1 e 2.

A.1 Grupo 1

Conj. Dados Cenário	Laplace Inicial – Final (Variação%)	+/-	#Iterações
breast			
Cnr1	0.9575 – 0.9369 (-002.15%) 0.0064 – 0.0359 (+003.61%)	+5-5	21 0
Cnr2	0.9579 – 0.9409 (-001.78%) 0.0042 – 0.0387 (+003.92%)	+6-4	21 0
Cnr3	0.9574 – 0.9580 (+000.06%) 0.0069 – 0.0072 (+000.37%)	+8-2	21 0
Cnr4	0.9576 – 0.9490 (-000.90%) 0.0059 – 0.0263 (+002.54%)	+8-2	21 0
Cnr5	0.9572 – 0.8966 (-006.31%) 0.0075 – 0.0242 (+003.03%)	-10	22 1
Cnr6	0.9575 – 0.9132 (-004.63%) 0.0064 – 0.0536 (+005.47%)	+4-6	21 2

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Laplace Inicial – Final (Variação%)	+/-	#Iterações
Cnr7	0.9574 – 0.9380 (-002.05%) 0.0069 – 0.0506 (+004.83%)	+7-3	21 0
Cnr8	0.9576 – 0.9338 (-002.49%) 0.0059 – 0.0305 (+003.07%)	+5-5	21 0
cmc			
Cnr1	0.6988 – 0.6846 (-001.41%) 0.0297 – 0.1430 (+023.28%)	+5-5	37 9
Cnr2	0.6715 – 0.7003 (+004.16%) 0.0151 – 0.1562 (+022.43%)	+6-4	32 6
Cnr3	0.7004 – 0.7977 (+014.10%) 0.0404 – 0.0775 (+011.17%)	+9-1	39 11
Cnr4	0.7108 – 0.6506 (-008.10%) 0.0375 – 0.1552 (+022.87%)	+4-6	29 6
Cnr5	0.6997 – 0.7417 (+006.20%) 0.0337 – 0.0637 (+010.20%)	+8-2	27 5
Cnr6	0.6881 – 0.7370 (+007.48%) 0.0358 – 0.0997 (+016.46%)	+8-2	28 6
Cnr7	0.6988 – 0.7963 (+014.12%) 0.0297 – 0.1174 (+017.36%)	+7-3	30 9
Cnr8	0.6715 – 0.6971 (+003.83%) 0.0151 – 0.1078 (+016.11%)	+7-3	29 5
heart			
Cnr1	0.9376 – 0.9050 (-003.46%) 0.0108 – 0.0223 (+002.65%)	+1-9	22 2
Cnr2	0.9376 – 0.8920 (-004.86%) 0.0108 – 0.0303 (+003.12%)	+1-9	22 2
Cnr3	0.9393 – 0.8986 (-004.34%) 0.0087 – 0.0355 (+003.64%)	-10	21 1
Cnr4	0.9407 – 0.8905 (-005.34%) 0.0127 – 0.0400 (+003.82%)	-10	22 2
Cnr5	0.9376 – 0.8906 (-005.00%) 0.0108 – 0.0355 (+004.00%)	+1-9	21 1
Cnr6	0.9385 – 0.8675 (-007.55%) 0.0084 – 0.0488 (+005.42%)	-10	22 2
Cnr7	0.9423 – 0.8936 (-005.17%) 0.0093 – 0.0293 (+003.02%)	+1-9	21 1
Cnr8	0.9423 – 0.8752 (-007.11%) 0.0093 – 0.0238 (+002.48%)	-10	21 2
nursery			
Cnr1	0.9945 – 0.9982 (+000.37%) 0.0001 – 0.0000 (+000.01%)	+10	21 0
Cnr2	0.9945 – 0.9982 (+000.37%) 0.0002 – 0.0000 (+000.02%)	+10	21 0
Cnr3	0.9945 – 0.9982 (+000.37%) 0.0001 – 0.0000 (+000.01%)	+10	21 0
Cnr4	0.9945 – 0.9982 (+000.38%) 0.0001 – 0.0000 (+000.01%)	+10	21 0
Cnr5	0.9945 – 0.9982 (+000.37%) 0.0001 – 0.0000 (+000.01%)	+10	21 0
Cnr6	0.9945 – 0.9982 (+000.37%) 0.0001 – 0.0000 (+000.01%)	+10	21 0
Cnr7	0.9945 – 0.9982 (+000.38%) 0.0001 – 0.0000 (+000.01%)	+10	21 0
Cnr8	0.9945 – 0.9982 (+000.37%) 0.0001 – 0.0000 (+000.01%)	+10	21 0

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Laplace Inicial – Final (Variação%)	+/-	#Iterações
spambase			
Cnr1	0.9971 – 0.9973 (+000.02%) 0.0001 – 0.0001 (+000.00%)	+10	21 0
Cnr2	0.9971 – 0.9973 (+000.02%) 0.0001 – 0.0001 (+000.01%)	+10	21 0
Cnr3	0.9971 – 0.9968 (-000.02%) 0.0001 – 0.0010 (+000.09%)	+8-2	21 0
Cnr4	0.9970 – 0.9969 (-000.02%) 0.0001 – 0.0009 (+000.09%)	+8-2	21 0
Cnr5	0.9971 – 0.9960 (-000.11%) 0.0001 – 0.0013 (+000.14%)	+5-5	21 0
Cnr6	0.9971 – 0.9957 (-000.13%) 0.0001 – 0.0017 (+000.18%)	+5-5	21 0
Cnr7	0.9970 – 0.9955 (-000.15%) 0.0001 – 0.0016 (+000.17%)	+4-6	21 0
Cnr8	0.9970 – 0.9957 (-000.14%) 0.0001 – 0.0019 (+000.18%)	+5-5	21 0
vehicle			
Cnr1	0.9303 – 0.9141 (-001.73%) 0.0121 – 0.0151 (+002.06%)	+5-5	21 1
Cnr2	0.9303 – 0.9195 (-001.15%) 0.0121 – 0.0087 (+001.40%)	+5-5	21 1
Cnr3	0.9305 – 0.9252 (-000.57%) 0.0135 – 0.0167 (+000.95%)	+5-5	21 0
Cnr4	0.9305 – 0.9245 (-000.64%) 0.0107 – 0.0196 (+001.83%)	+8-2	21 0
Cnr5	0.9300 – 0.9151 (-001.59%) 0.0130 – 0.0127 (+001.09%)	-10	22 1
Cnr6	0.9305 – 0.9186 (-001.27%) 0.0179 – 0.0187 (+001.40%)	+3-7	21 1
Cnr7	0.9304 – 0.9234 (-000.75%) 0.0077 – 0.0155 (+001.64%)	+8-2	21 0
Cnr8	0.9305 – 0.9214 (-000.97%) 0.0107 – 0.0205 (+002.04%)	+8-2	21 0
wbc			
Cnr1	0.9948 – 0.9918 (-000.30%) 0.0001 – 0.0034 (+000.34%)	+4-6	21 0
Cnr2	0.9948 – 0.9917 (-000.32%) 0.0001 – 0.0033 (+000.34%)	+2-8	21 0
Cnr3	0.9948 – 0.9929 (-000.19%) 0.0001 – 0.0047 (+000.47%)	+8-2	21 0
Cnr4	0.9948 – 0.9932 (-000.16%) 0.0001 – 0.0024 (+000.25%)	+4-6	21 0
Cnr5	0.9948 – 0.9908 (-000.40%) 0.0001 – 0.0053 (+000.53%)	+5-5	21 0
Cnr6	0.9948 – 0.9862 (-000.87%) 0.0001 – 0.0070 (+000.70%)	+1-9	21 0
Cnr7	0.9948 – 0.9928 (-000.20%) 0.0001 – 0.0026 (+000.26%)	+5-5	21 0
Cnr8	0.9948 – 0.9918 (-000.30%) 0.0001 – 0.0042 (+000.42%)	+5-5	21 0

Tabela A.1: Grupo 1 – Função de avaliação simples-objetivo – *Fnc1a*

Conj. Dados Cenário	Suporte Inicial – Final (Variação%)	+/-	#Iterações

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Suporte Inicial – Final (Variação%)	+/-	#Iterações
breast			
Cnr1	0.5326 – 0.7087 (+033.24%) 0.0209 – 0.0046 (+005.47%)	+10	27 2
Cnr2	0.5326 – 0.6348 (+019.04%) 0.0209 – 0.0649 (+009.37%)	+10	31 12
Cnr3	0.5326 – 0.6978 (+031.05%) 0.0078 – 0.0342 (+006.85%)	+10	24 1
Cnr4	0.5326 – 0.6811 (+027.97%) 0.0267 – 0.0380 (+005.86%)	+10	44 8
Cnr5	0.5326 – 0.7065 (+032.82%) 0.0214 – 0.0038 (+004.90%)	+10	24 1
Cnr6	0.5326 – 0.6587 (+023.89%) 0.0219 – 0.0151 (+006.38%)	+10	36 0
Cnr7	0.5326 – 0.7101 (+033.51%) 0.0209 – 0.0000 (+005.25%)	+10	27 2
Cnr8	0.5326 – 0.7087 (+033.33%) 0.0249 – 0.0046 (+006.67%)	+10	51 11
cmc			
Cnr1	0.1405 – 0.2862 (+103.40%) 0.0086 – 0.0971 (+067.02%)	+10	25 3
Cnr2	0.1406 – 0.2229 (+058.80%) 0.0048 – 0.0036 (+006.06%)	+10	29 3
Cnr3	0.1405 – 0.2460 (+075.64%) 0.0072 – 0.0635 (+047.17%)	+10	22 1
Cnr4	0.1405 – 0.2062 (+047.24%) 0.0067 – 0.0764 (+056.82%)	+10	26 4
Cnr5	0.1405 – 0.2261 (+060.96%) 0.0037 – 0.0009 (+004.16%)	+10	22 0
Cnr6	0.1405 – 0.2193 (+056.34%) 0.0060 – 0.0023 (+007.84%)	+10	22 0
Cnr7	0.1405 – 0.2261 (+061.38%) 0.0086 – 0.0009 (+009.89%)	+10	24 0
Cnr8	0.1405 – 0.2193 (+056.44%) 0.0072 – 0.0022 (+008.86%)	+10	27 0
heart			
Cnr1	0.3444 – 0.5556 (+061.59%) 0.0154 – 0.0000 (+007.21%)	+10	24 1
Cnr2	0.3444 – 0.5526 (+060.76%) 0.0154 – 0.0072 (+008.37%)	+10	25 2
Cnr3	0.3444 – 0.5526 (+061.75%) 0.0326 – 0.0052 (+015.66%)	+10	28 4
Cnr4	0.3444 – 0.5437 (+058.86%) 0.0282 – 0.0232 (+015.54%)	+10	31 5
Cnr5	0.3444 – 0.5504 (+060.05%) 0.0154 – 0.0116 (+007.27%)	+10	25 2
Cnr6	0.3444 – 0.4882 (+042.02%) 0.0189 – 0.0558 (+017.38%)	+10	29 6
Cnr7	0.3444 – 0.5556 (+061.59%) 0.0154 – 0.0000 (+007.21%)	+10	25 0
Cnr8	0.3444 – 0.5392 (+057.46%) 0.0282 – 0.0098 (+012.46%)	+10	33 1
nursery			
Cnr1	0.1111 – 0.3333 (+200.11%) 0.0025 – 0.0000 (+006.73%)	+10	21 0
Cnr2	0.1111 – 0.3333 (+200.18%) 0.0033 – 0.0000 (+008.91%)	+10	22 0

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Suporte Inicial – Final (Variação%)	+/-	#Iterações
Cnr3	0.1111 – 0.3333 (+200.23%) 0.0032 – 0.0000 (+008.75%)	+10	21 0
Cnr4	0.1111 – 0.3333 (+200.05%) 0.0015 – 0.0000 (+004.16%)	+10	21 0
Cnr5	0.1111 – 0.3333 (+200.06%) 0.0016 – 0.0000 (+004.31%)	+10	21 0
Cnr6	0.1111 – 0.3333 (+200.08%) 0.0019 – 0.0000 (+005.16%)	+10	21 0
Cnr7	0.1111 – 0.3333 (+200.06%) 0.0020 – 0.0000 (+005.40%)	+10	21 0
Cnr8	0.1111 – 0.3333 (+200.23%) 0.0032 – 0.0000 (+008.75%)	+10	24 1
spambase			
Cnr1	0.5884 – 0.6059 (+002.97%) 0.0024 – 0.0002 (+000.41%)	+10	21 0
Cnr2	0.5884 – 0.6059 (+002.99%) 0.0024 – 0.0002 (+000.42%)	+10	21 0
Cnr3	0.5884 – 0.6058 (+002.96%) 0.0024 – 0.0003 (+000.44%)	+10	21 0
Cnr4	0.5887 – 0.6059 (+002.93%) 0.0014 – 0.0002 (+000.25%)	+10	21 0
Cnr5	0.5884 – 0.6059 (+002.99%) 0.0029 – 0.0002 (+000.51%)	+10	21 0
Cnr6	0.5885 – 0.6059 (+002.96%) 0.0022 – 0.0002 (+000.39%)	+10	21 0
Cnr7	0.5883 – 0.6058 (+002.97%) 0.0011 – 0.0005 (+000.20%)	+10	21 0
Cnr8	0.5883 – 0.6059 (+002.98%) 0.0011 – 0.0002 (+000.18%)	+10	21 0
vehicle			
Cnr1	0.1241 – 0.2369 (+092.35%) 0.0112 – 0.0050 (+019.12%)	+10	22 0
Cnr2	0.1241 – 0.2369 (+092.35%) 0.0112 – 0.0050 (+019.12%)	+10	22 0
Cnr3	0.1241 – 0.2371 (+091.75%) 0.0085 – 0.0065 (+012.71%)	+10	25 1
Cnr4	0.1241 – 0.2220 (+079.71%) 0.0136 – 0.0171 (+012.47%)	+10	32 7
Cnr5	0.1241 – 0.2553 (+107.21%) 0.0109 – 0.0022 (+018.98%)	+10	23 1
Cnr6	0.1241 – 0.2537 (+105.06%) 0.0077 – 0.0030 (+012.59%)	+10	27 6
Cnr7	0.1241 – 0.2501 (+102.38%) 0.0073 – 0.0106 (+018.14%)	+10	25 1
Cnr8	0.1241 – 0.2149 (+073.62%) 0.0112 – 0.0221 (+016.64%)	+10	30 5
wbc			
Cnr1	0.5623 – 0.6500 (+015.63%) 0.0065 – 0.0010 (+001.33%)	+10	21 0
Cnr2	0.5622 – 0.6500 (+015.64%) 0.0072 – 0.0010 (+001.51%)	+10	21 0
Cnr3	0.5622 – 0.6500 (+015.63%) 0.0065 – 0.0010 (+001.24%)	+10	24 3
Cnr4	0.5622 – 0.6492 (+015.48%) 0.0064 – 0.0017 (+001.33%)	+10	30 5

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Suporte Inicial – Final (Variação%)	+/-	#Iterações
Cnr5	0.5631 – 0.6500 (+015.47%) 0.0101 – 0.0010 (+001.95%)	+10	21 0
Cnr6	0.5622 – 0.6448 (+014.72%) 0.0093 – 0.0056 (+002.24%)	+10	21 0
Cnr7	0.5628 – 0.6500 (+015.53%) 0.0088 – 0.0010 (+001.72%)	+10	22 0
Cnr8	0.5634 – 0.6179 (+009.69%) 0.0135 – 0.0089 (+001.55%)	+10	21 0

Tabela A.2: Grupo 1 – Função de avaliação simples-objetivo – *Fnc1b*

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	+/-	#Iterações
breast			
Cnr1	0.0721 – 0.0664 (-004.84%) 0.0122 – 0.0159 (+026.85%)	+6-4	21 0
Cnr2	0.0720 – 0.0679 (-004.45%) 0.0097 – 0.0060 (+013.15%)	+4-6	21 0
Cnr3	0.0735 – 0.0790 (+010.30%) 0.0172 – 0.0132 (+019.30%)	+7-3	21 0
Cnr4	0.0731 – 0.0663 (-008.37%) 0.0199 – 0.0160 (+011.61%)	+4-6	21 0
Cnr5	0.0721 – 0.0784 (+011.37%) 0.0214 – 0.0182 (+015.00%)	+7-3	21 0
Cnr6	0.0720 – 0.0692 (-003.27%) 0.0109 – 0.0104 (+011.01%)	+4-6	22 2
Cnr7	0.0720 – 0.0711 (+000.65%) 0.0146 – 0.0150 (+020.42%)	+6-4	22 2
Cnr8	0.0720 – 0.0607 (-012.46%) 0.0146 – 0.0147 (+026.08%)	+3-7	21 1
cmc			
Cnr1	0.0529 – 0.0572 (+007.98%) 0.0078 – 0.0112 (+011.74%)	+10	21 0
Cnr2	0.0529 – 0.0531 (+000.49%) 0.0052 – 0.0049 (+001.18%)	+10	21 0
Cnr3	0.0529 – 0.0548 (+003.43%) 0.0070 – 0.0083 (+003.98%)	+10	21 0
Cnr4	0.0529 – 0.0521 (-001.49%) 0.0040 – 0.0060 (+009.02%)	+9-1	21 0
Cnr5	0.0529 – 0.0554 (+004.98%) 0.0029 – 0.0023 (+005.62%)	+9-1	21 0
Cnr6	0.0529 – 0.0500 (-005.13%) 0.0029 – 0.0085 (+016.52%)	+7-3	21 0
Cnr7	0.0529 – 0.0537 (+001.58%) 0.0078 – 0.0072 (+002.24%)	+10	21 0
Cnr8	0.0529 – 0.0527 (-000.55%) 0.0070 – 0.0074 (+001.75%)	+9-1	21 0
heart			
Cnr1	0.1366 – 0.1328 (-002.74%) 0.0076 – 0.0095 (+005.84%)	+5-5	21 0
Cnr2	0.1379 – 0.1335 (-003.09%) 0.0061 – 0.0067 (+004.48%)	+5-5	21 1
Cnr3	0.1369 – 0.1273 (-006.37%) 0.0162 – 0.0140 (+010.08%)	+3-7	21 1
Cnr4	0.1366 – 0.1313 (-003.59%) 0.0138 – 0.0147 (+009.29%)	+4-6	21 0

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr5	0.1379 – 0.1256 (-008.62%) 0.0061 – 0.0163 (+012.69%)	+4-6	21 0
Cnr6	0.1370 – 0.1302 (-004.84%) 0.0079 – 0.0091 (+005.57%)	+5-5	21 0
Cnr7	0.1379 – 0.1254 (-008.66%) 0.0061 – 0.0178 (+014.47%)	+5-5	21 0
Cnr8	0.1366 – 0.1277 (-006.33%) 0.0138 – 0.0151 (+007.94%)	+5-5	21 0
nursery			
Cnr1	0.0741 – 0.2222 (+200.30%) 0.0022 – 0.0000 (+008.88%)	+10	21 0
Cnr2	0.0741 – 0.2222 (+199.96%) 0.0010 – 0.0000 (+004.25%)	+10	22 0
Cnr3	0.0741 – 0.2222 (+200.14%) 0.0015 – 0.0000 (+006.01%)	+10	21 0
Cnr4	0.0741 – 0.2222 (+199.96%) 0.0010 – 0.0000 (+004.25%)	+10	21 0
Cnr5	0.0741 – 0.2222 (+200.14%) 0.0017 – 0.0000 (+007.09%)	+10	21 0
Cnr6	0.0741 – 0.2222 (+200.07%) 0.0013 – 0.0000 (+005.16%)	+10	21 0
Cnr7	0.0741 – 0.2222 (+200.30%) 0.0022 – 0.0000 (+008.88%)	+10	21 0
Cnr8	0.0741 – 0.2222 (+200.17%) 0.0022 – 0.0000 (+008.73%)	+10	24 1
spambase			
Cnr1	0.1189 – 0.1654 (+039.19%) 0.0029 – 0.0079 (+007.68%)	+10	22 2
Cnr2	0.1189 – 0.1665 (+040.11%) 0.0027 – 0.0098 (+008.92%)	+10	23 3
Cnr3	0.1189 – 0.1655 (+039.24%) 0.0029 – 0.0088 (+007.62%)	+10	23 2
Cnr4	0.1189 – 0.1741 (+046.52%) 0.0029 – 0.0080 (+008.02%)	+10	25 2
Cnr5	0.1189 – 0.1774 (+049.21%) 0.0023 – 0.0031 (+002.19%)	+10	21 0
Cnr6	0.1189 – 0.1769 (+048.85%) 0.0035 – 0.0024 (+003.23%)	+10	22 0
Cnr7	0.1189 – 0.1493 (+025.67%) 0.0024 – 0.0053 (+005.38%)	+10	22 0
Cnr8	0.1189 – 0.1523 (+028.16%) 0.0017 – 0.0019 (+001.96%)	+10	23 0
vehicle			
Cnr1	0.0944 – 0.1133 (+020.20%) 0.0051 – 0.0068 (+005.63%)	+10	21 0
Cnr2	0.0944 – 0.1116 (+019.05%) 0.0085 – 0.0069 (+011.56%)	+9-1	21 0
Cnr3	0.0944 – 0.1119 (+019.19%) 0.0063 – 0.0050 (+010.65%)	+9-1	22 0
Cnr4	0.0944 – 0.1068 (+013.37%) 0.0107 – 0.0134 (+008.93%)	+9-1	21 0
Cnr5	0.0944 – 0.1069 (+014.06%) 0.0087 – 0.0065 (+011.54%)	+9-1	21 0
Cnr6	0.0944 – 0.1107 (+017.51%) 0.0058 – 0.0054 (+005.87%)	+10	21 0

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr7	0.0944 – 0.1076 (+014.09%) 0.0047 – 0.0062 (+006.18%)	+10	21 0
Cnr8	0.0944 – 0.1078 (+014.77%) 0.0086 – 0.0049 (+007.63%)	+9-1	23 5
wbc			
Cnr1	0.1967 – 0.2092 (+006.37%) 0.0023 – 0.0036 (+002.04%)	+10	21 0
Cnr2	0.1967 – 0.2077 (+005.56%) 0.0026 – 0.0044 (+001.13%)	+10	21 0
Cnr3	0.1967 – 0.2076 (+005.57%) 0.0041 – 0.0024 (+002.55%)	+10	21 0
Cnr4	0.1967 – 0.2076 (+005.54%) 0.0024 – 0.0038 (+002.84%)	+10	21 0
Cnr5	0.1967 – 0.2073 (+005.39%) 0.0034 – 0.0018 (+001.49%)	+10	21 0
Cnr6	0.1967 – 0.2055 (+004.47%) 0.0023 – 0.0049 (+002.47%)	+10	21 0
Cnr7	0.1970 – 0.2063 (+004.69%) 0.0032 – 0.0046 (+001.53%)	+10	21 0
Cnr8	0.1972 – 0.2069 (+004.98%) 0.0047 – 0.0066 (+003.13%)	+10	21 0

Tabela A.3: Grupo 1 – Função de avaliação simples-objetivo – *Fnc1c*

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	+/-	#Iterações
breast			
Cnr1	0.7500 – 0.9980 (+033.24%) 0.0293 – 0.0043 (+005.18%)	+10	28 2
Cnr2	0.7500 – 0.8979 (+019.86%) 0.0260 – 0.0795 (+011.51%)	+10	29 9
Cnr3	0.7500 – 0.9990 (+033.40%) 0.0301 – 0.0032 (+005.51%)	+10	30 5
Cnr4	0.7500 – 0.9368 (+024.97%) 0.0301 – 0.0565 (+007.06%)	+10	38 12
Cnr5	0.7500 – 0.9980 (+033.28%) 0.0323 – 0.0043 (+005.72%)	+10	25 1
Cnr6	0.7500 – 0.9204 (+022.98%) 0.0376 – 0.0262 (+006.69%)	+10	36 0
Cnr7	0.7500 – 0.9990 (+033.46%) 0.0351 – 0.0032 (+006.27%)	+10	27 2
Cnr8	0.7500 – 0.9980 (+033.20%) 0.0251 – 0.0065 (+004.83%)	+10	45 14
cmc			
Cnr1	0.6216 – 1.0000 (+061.41%) 0.0381 – 0.0000 (+009.91%)	+10	24 1
Cnr2	0.6216 – 0.9491 (+052.61%) 0.0104 – 0.1198 (+018.75%)	+10	29 6
Cnr3	0.6217 – 1.0000 (+061.46%) 0.0400 – 0.0000 (+010.48%)	+10	23 0
Cnr4	0.6217 – 0.8013 (+029.06%) 0.0183 – 0.1887 (+031.06%)	+10	26 6
Cnr5	0.6217 – 1.0000 (+061.14%) 0.0273 – 0.0000 (+007.09%)	+10	25 1
Cnr6	0.6216 – 0.9700 (+056.19%) 0.0199 – 0.0085 (+005.35%)	+10	22 0

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr7	0.6216 – 1.0000 (+061.41%) 0.0381 – 0.0000 (+009.91%)	+10	24 0
Cnr8	0.6217 – 0.9700 (+056.57%) 0.0400 – 0.0098 (+009.49%)	+10	27 0
heart			
Cnr1	0.6200 – 0.9967 (+061.07%) 0.0276 – 0.0070 (+007.90%)	+10	24 1
Cnr2	0.6243 – 0.9907 (+058.98%) 0.0299 – 0.0110 (+007.23%)	+10	25 2
Cnr3	0.6200 – 0.9960 (+061.60%) 0.0516 – 0.0090 (+012.86%)	+10	27 3
Cnr4	0.6200 – 0.9693 (+056.40%) 0.0211 – 0.0741 (+011.93%)	+10	30 7
Cnr5	0.6243 – 0.9973 (+060.09%) 0.0299 – 0.0056 (+008.17%)	+10	26 2
Cnr6	0.6243 – 0.8683 (+039.46%) 0.0299 – 0.0882 (+016.75%)	+10	27 4
Cnr7	0.6200 – 0.9960 (+061.34%) 0.0432 – 0.0090 (+011.15%)	+10	27 1
Cnr8	0.6200 – 0.9707 (+056.69%) 0.0211 – 0.0225 (+005.71%)	+10	33 1
nursery			
Cnr1	0.3333 – 1.0000 (+200.12%) 0.0072 – 0.0000 (+006.51%)	+10	21 0
Cnr2	0.3333 – 1.0000 (+200.07%) 0.0057 – 0.0000 (+005.15%)	+10	21 0
Cnr3	0.3333 – 1.0000 (+200.11%) 0.0066 – 0.0000 (+005.98%)	+10	21 0
Cnr4	0.3333 – 1.0000 (+200.23%) 0.0099 – 0.0000 (+008.93%)	+10	21 0
Cnr5	0.3333 – 1.0000 (+200.06%) 0.0047 – 0.0000 (+004.28%)	+10	21 0
Cnr6	0.3333 – 1.0000 (+200.18%) 0.0088 – 0.0000 (+007.97%)	+10	21 0
Cnr7	0.3333 – 1.0000 (+200.12%) 0.0072 – 0.0000 (+006.51%)	+10	21 0
Cnr8	0.3333 – 1.0000 (+200.11%) 0.0066 – 0.0000 (+005.98%)	+10	24 1
spambase			
Cnr1	0.9710 – 1.0000 (+002.99%) 0.0028 – 0.0000 (+000.30%)	+10	21 0
Cnr2	0.9710 – 1.0000 (+002.99%) 0.0028 – 0.0000 (+000.30%)	+10	21 0
Cnr3	0.9710 – 0.9999 (+002.98%) 0.0028 – 0.0003 (+000.31%)	+10	21 0
Cnr4	0.9709 – 0.9999 (+002.98%) 0.0008 – 0.0002 (+000.08%)	+10	21 0
Cnr5	0.9709 – 1.0000 (+002.99%) 0.0040 – 0.0000 (+000.42%)	+10	21 0
Cnr6	0.9709 – 1.0000 (+002.99%) 0.0040 – 0.0000 (+000.42%)	+10	21 0
Cnr7	0.9709 – 1.0000 (+002.99%) 0.0019 – 0.0000 (+000.20%)	+10	21 0
Cnr8	0.9709 – 1.0000 (+002.99%) 0.0019 – 0.0000 (+000.20%)	+10	21 0
vehicle			

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr1	0.5275 – 1.0000 (+090.88%) 0.0456 – 0.0000 (+016.69%)	+10	23 1
Cnr2	0.5275 – 1.0000 (+090.28%) 0.0342 – 0.0000 (+012.49%)	+10	23 0
Cnr3	0.5276 – 0.9960 (+089.22%) 0.0269 – 0.0085 (+009.95%)	+10	28 2
Cnr4	0.5277 – 0.9560 (+081.41%) 0.0319 – 0.0905 (+016.76%)	+10	38 8
Cnr5	0.5276 – 1.0000 (+090.94%) 0.0473 – 0.0000 (+017.69%)	+10	23 0
Cnr6	0.5276 – 1.0000 (+089.98%) 0.0269 – 0.0000 (+009.78%)	+10	27 11
Cnr7	0.5276 – 1.0000 (+090.09%) 0.0300 – 0.0000 (+010.86%)	+10	26 1
Cnr8	0.5276 – 0.9396 (+078.57%) 0.0271 – 0.0798 (+018.39%)	+10	33 7
wbc			
Cnr1	0.8649 – 1.0000 (+015.63%) 0.0099 – 0.0000 (+001.33%)	+10	22 0
Cnr2	0.8649 – 0.9995 (+015.59%) 0.0112 – 0.0014 (+001.47%)	+10	23 2
Cnr3	0.8649 – 1.0000 (+015.67%) 0.0188 – 0.0000 (+002.51%)	+10	26 1
Cnr4	0.8658 – 0.9914 (+014.55%) 0.0129 – 0.0188 (+003.34%)	+10	33 7
Cnr5	0.8649 – 1.0000 (+015.64%) 0.0102 – 0.0000 (+001.36%)	+10	21 0
Cnr6	0.8658 – 0.9910 (+014.48%) 0.0129 – 0.0092 (+002.04%)	+10	21 0
Cnr7	0.8662 – 1.0000 (+015.47%) 0.0147 – 0.0000 (+001.95%)	+10	23 1
Cnr8	0.8667 – 0.9505 (+009.69%) 0.0206 – 0.0140 (+001.55%)	+10	21 0

Tabela A.4: Grupo 1 – Função de avaliação simples-objetivo – *Fnc2a*

Conj. Dados Cenário	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
breast – cmc – heart – nursery – spambase			
Cnr1	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr2	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr3	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr4	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr5	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr6	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr7	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr8	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
vehicle			

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr1	1.0000 – 0.9990 (-000.10%) 0.0000 – 0.0030 (+000.30%)	+9-1	21 0
Cnr2	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr3	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr4	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr5	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr6	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr7	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr8	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
wbc			
Cnr1	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr2	1.0000 – 0.9995 (-000.04%) 0.0000 – 0.0014 (+000.14%)	+9-1	21 0
Cnr3	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr4	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr5	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr6	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr7	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr8	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0

Tabela A.5: Grupo 1 – Função de avaliação simples-objetivo – *Fnc2b*

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	Laplace Inicial – Final (Variação%)	Suporte Inicial – Final (Variação%)	+/-	#Iterações
breast					
Cnr1	0.0490 – 0.0614 (+029.69%) 0.0140 – 0.0172 (+041.68%)	0.9448 – 0.8027 (-014.90%) 0.0376 – 0.0287 (+005.04%)	0.1920 – 0.5029 (+205.66%) 0.1179 – 0.0754 (+096.28%)	+7-3	29 8
Cnr2	0.0505 – 0.0689 (+041.08%) 0.0131 – 0.0134 (+032.16%)	0.9470 – 0.8177 (-013.51%) 0.0360 – 0.0279 (+004.94%)	0.1956 – 0.5022 (+195.38%) 0.1107 – 0.0672 (+089.47%)	+9-1	27 6
Cnr3	0.0505 – 0.0720 (+047.08%) 0.0131 – 0.0154 (+036.72%)	0.9470 – 0.8217 (-013.08%) 0.0360 – 0.0329 (+005.62%)	0.1956 – 0.5130 (+196.52%) 0.1107 – 0.0546 (+069.75%)	+9-1	32 8
Cnr4	0.0541 – 0.0585 (+012.76%) 0.0148 – 0.0104 (+025.27%)	0.9154 – 0.8272 (-009.21%) 0.0678 – 0.0356 (+007.43%)	0.2703 – 0.4094 (+099.27%) 0.1835 – 0.1110 (+099.80%)	+8-2	24 2
Cnr5	0.0541 – 0.0715 (+038.68%) 0.0148 – 0.0090 (+031.39%)	0.9154 – 0.8260 (-009.29%) 0.0678 – 0.0224 (+007.69%)	0.2703 – 0.4862 (+145.81%) 0.1835 – 0.0415 (+112.48%)	+8-2	22 2
Cnr6	0.0496 – 0.0688 (+041.73%) 0.0095 – 0.0144 (+034.98%)	0.9444 – 0.8159 (-013.42%) 0.0454 – 0.0298 (+005.22%)	0.1978 – 0.5072 (+193.63%) 0.1108 – 0.0343 (+084.13%)	+9-1	25 4
Cnr7	0.0496 – 0.0679 (+041.32%) 0.0095 – 0.0117 (+037.55%)	0.9444 – 0.8094 (-014.12%) 0.0454 – 0.0202 (+004.62%)	0.1978 – 0.5254 (+203.47%) 0.1108 – 0.0474 (+085.98%)	+9-1	21 0
Cnr8	0.0505 – 0.0798 (+064.55%) 0.0131 – 0.0080 (+031.28%)	0.9470 – 0.8432 (-010.79%) 0.0360 – 0.0199 (+005.45%)	0.1956 – 0.4848 (+187.30%) 0.1107 – 0.0548 (+079.32%)	+9-1	25 10
cmc					

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	Laplace Inicial – Final (Variação%)	Suporte Inicial – Final (Variação%)	+/-	#Iterações
Cnr1	0.0052 – 0.0534 (+1203.09%)	0.6743 – 0.4222 (-036.79%)	0.0126 – 0.1889 (+1954.44%)	+9-1	29
	0.0026 – 0.0173 (+621.01%)	0.0597 – 0.0478 (+010.30%)	0.0095 – 0.0843 (+991.27%)		9
Cnr2	0.0052 – 0.0518 (+1044.65%)	0.6743 – 0.4109 (-038.43%)	0.0126 – 0.1776 (+1884.81%)	+10	25
	0.0026 – 0.0123 (+376.03%)	0.0597 – 0.0500 (+010.40%)	0.0095 – 0.0761 (+1313.54%)		5
Cnr3	0.0098 – 0.0474 (+919.87%)	0.6584 – 0.3928 (-037.89%)	0.0238 – 0.1659 (+2468.27%)	+10	36
	0.0157 – 0.0167 (+679.41%)	0.1169 – 0.0374 (+016.65%)	0.0420 – 0.0765 (+2632.70%)		23
Cnr4	0.0060 – 0.0525 (+1063.78%)	0.6916 – 0.3745 (-045.60%)	0.0138 – 0.1325 (+1519.26%)	+10	22
	0.0037 – 0.0087 (+614.95%)	0.0456 – 0.0262 (+005.61%)	0.0110 – 0.0156 (+1026.41%)		3
Cnr5	0.0060 – 0.0459 (+981.12%)	0.6916 – 0.4282 (-037.79%)	0.0138 – 0.1775 (+1695.40%)	+10	39
	0.0037 – 0.0212 (+812.47%)	0.0456 – 0.0908 (+013.63%)	0.0110 – 0.1090 (+999.28%)		30
Cnr6	0.0056 – 0.0505 (+1027.30%)	0.6570 – 0.3627 (-044.40%)	0.0147 – 0.1333 (+1492.60%)	+10	21
	0.0028 – 0.0086 (+597.11%)	0.0566 – 0.0170 (+005.89%)	0.0102 – 0.0159 (+1159.06%)		0
Cnr7	0.0098 – 0.0569 (+1225.12%)	0.6584 – 0.3810 (-039.87%)	0.0238 – 0.1401 (+1846.93%)	+10	21
	0.0157 – 0.0034 (+730.24%)	0.1169 – 0.0161 (+014.82%)	0.0420 – 0.0022 (+1247.41%)		0
Cnr8	0.0097 – 0.0521 (+1233.85%)	0.6298 – 0.3631 (-040.20%)	0.0247 – 0.1381 (+2017.71%)	+9-1	21
	0.0155 – 0.0045 (+793.25%)	0.0999 – 0.0163 (+015.11%)	0.0428 – 0.0042 (+1481.81%)		0
heart					
Cnr1	0.1366 – 0.1360 (-000.44%)	0.9053 – 0.8717 (-003.73%)	0.3444 – 0.3726 (+008.51%)	+6-4	21
	0.0076 – 0.0092 (+004.92%)	0.0225 – 0.0622 (+006.11%)	0.0154 – 0.0381 (+013.76%)		0
Cnr2	0.1366 – 0.1342 (-001.68%)	0.9053 – 0.8741 (-003.45%)	0.3444 – 0.3622 (+005.21%)	+7-3	21
	0.0076 – 0.0079 (+004.44%)	0.0225 – 0.0455 (+004.34%)	0.0154 – 0.0301 (+008.07%)		1
Cnr3	0.1253 – 0.1310 (+006.55%)	0.9117 – 0.8425 (-007.54%)	0.3000 – 0.3652 (+028.55%)	+6-4	23
	0.0204 – 0.0097 (+014.32%)	0.0158 – 0.0714 (+008.41%)	0.0705 – 0.0419 (+036.17%)		4
Cnr4	0.1345 – 0.1243 (-006.88%)	0.9050 – 0.8348 (-007.66%)	0.3318 – 0.3741 (+013.75%)	+5-5	21
	0.0151 – 0.0129 (+011.43%)	0.0190 – 0.0683 (+008.47%)	0.0477 – 0.0616 (+018.17%)		1
Cnr5	0.1359 – 0.1293 (-004.63%)	0.9103 – 0.8596 (-005.56%)	0.3333 – 0.3622 (+010.03%)	+5-5	21
	0.0086 – 0.0090 (+007.48%)	0.0188 – 0.0579 (+006.18%)	0.0406 – 0.0347 (+015.86%)		1
Cnr6	0.1291 – 0.1252 (+000.90%)	0.9171 – 0.8277 (-009.70%)	0.3126 – 0.3674 (+031.18%)	+5-5	21
	0.0253 – 0.0203 (+025.43%)	0.0247 – 0.1009 (+011.11%)	0.0726 – 0.0474 (+070.92%)		0
Cnr7	0.1298 – 0.1304 (+002.01%)	0.9195 – 0.8310 (-009.53%)	0.2926 – 0.3481 (+025.29%)	+6-4	22
	0.0135 – 0.0124 (+018.18%)	0.0231 – 0.0974 (+011.14%)	0.0629 – 0.0318 (+034.43%)		1
Cnr8	0.1307 – 0.1272 (+004.10%)	0.9153 – 0.8711 (-004.74%)	0.3059 – 0.3222 (+021.60%)	+5-5	21
	0.0265 – 0.0099 (+038.72%)	0.0287 – 0.0600 (+007.05%)	0.0843 – 0.0391 (+070.53%)		1
nursery					
Cnr1	0.0741 – 0.2222 (+200.08%)	0.9945 – 0.9982 (+000.37%)	0.1111 – 0.3333 (+200.11%)	+10	21
	0.0017 – 0.0000 (+006.79%)	0.0001 – 0.0000 (+000.01%)	0.0025 – 0.0000 (+006.73%)		0
Cnr2	0.0741 – 0.2222 (+200.30%)	0.9945 – 0.9982 (+000.37%)	0.1111 – 0.3333 (+200.18%)	+10	25
	0.0022 – 0.0000 (+008.88%)	0.0002 – 0.0000 (+000.02%)	0.0033 – 0.0000 (+008.91%)		5
Cnr3	0.0741 – 0.2222 (+200.14%)	0.9945 – 0.9982 (+000.37%)	0.1111 – 0.3333 (+200.08%)	+10	25
	0.0015 – 0.0000 (+006.01%)	0.0001 – 0.0000 (+000.01%)	0.0022 – 0.0000 (+005.98%)		2
Cnr4	0.0741 – 0.2222 (+199.96%)	0.9945 – 0.9982 (+000.38%)	0.1111 – 0.3333 (+200.05%)	+10	22
	0.0010 – 0.0000 (+004.08%)	0.0001 – 0.0000 (+000.01%)	0.0015 – 0.0000 (+004.16%)		2
Cnr5	0.0741 – 0.2222 (+200.10%)	0.9945 – 0.9982 (+000.37%)	0.1111 – 0.3333 (+200.02%)	+10	21
	0.0012 – 0.0000 (+005.00%)	0.0001 – 0.0000 (+000.01%)	0.0018 – 0.0000 (+004.95%)		0
Cnr6	0.0741 – 0.2222 (+200.07%)	0.9945 – 0.9982 (+000.37%)	0.1111 – 0.3333 (+200.08%)	+10	21
	0.0013 – 0.0000 (+005.16%)	0.0001 – 0.0000 (+000.01%)	0.0019 – 0.0000 (+005.16%)		0
Cnr7	0.0741 – 0.2222 (+200.14%)	0.9945 – 0.9982 (+000.37%)	0.1111 – 0.3333 (+200.14%)	+10	21
	0.0020 – 0.0000 (+007.98%)	0.0001 – 0.0000 (+000.01%)	0.0030 – 0.0000 (+008.00%)		0
Cnr8	0.0741 – 0.2222 (+200.07%)	0.9945 – 0.9982 (+000.37%)	0.1111 – 0.3333 (+200.06%)	+10	23
	0.0013 – 0.0000 (+005.41%)	0.0001 – 0.0000 (+000.01%)	0.0020 – 0.0000 (+005.40%)		1
spambase					
Cnr1	0.1167 – 0.1703 (+046.86%)	0.9394 – 0.8929 (-004.93%)	0.2098 – 0.5300 (+153.32%)	+10	24
	0.0100 – 0.0086 (+012.99%)	0.0096 – 0.0208 (+002.74%)	0.0121 – 0.0120 (+014.45%)		2
Cnr2	0.1137 – 0.1701 (+051.31%)	0.9431 – 0.8976 (-004.81%)	0.2132 – 0.5236 (+146.98%)	+10	23
	0.0121 – 0.0082 (+019.23%)	0.0146 – 0.0216 (+002.19%)	0.0161 – 0.0148 (+021.62%)		1

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	Laplace Inicial – Final (Variação%)	Suporte Inicial – Final (Variação%)	+/-	#Iterações
Cnr3	0.1130 – 0.1773 (+058.61%)	0.9458 – 0.9210 (-002.60%)	0.2115 – 0.5181 (+146.15%)	+10	26
	0.0113 – 0.0028 (+019.63%)	0.0162 – 0.0122 (+001.91%)	0.0174 – 0.0166 (+016.91%)		3
Cnr4	0.1137 – 0.1785 (+058.90%)	0.9431 – 0.9211 (-002.30%)	0.2132 – 0.5213 (+145.56%)	+10	25
	0.0121 – 0.0020 (+020.28%)	0.0146 – 0.0113 (+002.04%)	0.0161 – 0.0105 (+016.60%)		4
Cnr5	0.1161 – 0.1786 (+055.09%)	0.9403 – 0.9197 (-002.18%)	0.2083 – 0.5229 (+151.67%)	+10	22
	0.0100 – 0.0026 (+015.70%)	0.0123 – 0.0065 (+001.48%)	0.0108 – 0.0065 (+012.67%)		0
Cnr6	0.1114 – 0.1778 (+061.97%)	0.9499 – 0.9165 (-003.47%)	0.2168 – 0.5243 (+143.30%)	+10	22
	0.0139 – 0.0022 (+021.42%)	0.0186 – 0.0105 (+002.42%)	0.0178 – 0.0111 (+020.14%)		0
Cnr7	0.1114 – 0.1783 (+062.48%)	0.9499 – 0.9206 (-003.05%)	0.2168 – 0.5211 (+141.69%)	+10	23
	0.0139 – 0.0027 (+022.16%)	0.0186 – 0.0066 (+002.15%)	0.0178 – 0.0055 (+017.87%)		0
Cnr8	0.1159 – 0.1776 (+054.20%)	0.9408 – 0.9216 (-002.02%)	0.2079 – 0.5177 (+149.58%)	+10	25
	0.0097 – 0.0034 (+014.29%)	0.0141 – 0.0073 (+001.83%)	0.0100 – 0.0070 (+012.80%)		0
vehicle					
Cnr1	0.0944 – 0.1192 (+026.48%)	0.9303 – 0.5647 (-039.23%)	0.1241 – 0.2071 (+067.18%)	+10	29
	0.0051 – 0.0077 (+008.68%)	0.0121 – 0.0906 (+010.25%)	0.0066 – 0.0206 (+017.93%)		10
Cnr2	0.0944 – 0.1137 (+020.58%)	0.9303 – 0.5551 (-040.26%)	0.1241 – 0.2005 (+061.54%)	+10	26
	0.0051 – 0.0076 (+007.15%)	0.0121 – 0.1017 (+011.50%)	0.0066 – 0.0256 (+019.11%)		2
Cnr3	0.0944 – 0.1077 (+014.62%)	0.9305 – 0.5832 (-037.34%)	0.1241 – 0.1849 (+049.17%)	+9-1	30
	0.0063 – 0.0050 (+008.71%)	0.0135 – 0.1155 (+012.38%)	0.0085 – 0.0214 (+016.72%)		3
Cnr4	0.0944 – 0.1074 (+013.99%)	0.9306 – 0.6160 (-033.76%)	0.1241 – 0.1790 (+044.00%)	+10	32
	0.0049 – 0.0069 (+008.44%)	0.0097 – 0.1375 (+015.05%)	0.0064 – 0.0230 (+014.71%)		4
Cnr5	0.0944 – 0.1056 (+012.60%)	0.9300 – 0.5853 (-036.96%)	0.1241 – 0.1808 (+045.52%)	+9-1	29
	0.0086 – 0.0065 (+010.80%)	0.0130 – 0.1053 (+011.90%)	0.0112 – 0.0256 (+013.81%)		5
Cnr6	0.0944 – 0.1083 (+014.77%)	0.9305 – 0.5925 (-036.33%)	0.1241 – 0.1801 (+045.30%)	+10	36
	0.0058 – 0.0072 (+003.46%)	0.0179 – 0.0850 (+008.80%)	0.0077 – 0.0168 (+012.30%)		9
Cnr7	0.0944 – 0.1061 (+012.60%)	0.9309 – 0.6467 (-030.53%)	0.1241 – 0.1697 (+037.54%)	+9-1	25
	0.0047 – 0.0050 (+006.97%)	0.0139 – 0.1261 (+013.47%)	0.0066 – 0.0210 (+021.53%)		2
Cnr8	0.0944 – 0.1061 (+012.50%)	0.9309 – 0.6602 (-029.10%)	0.1241 – 0.1685 (+036.49%)	+10	30
	0.0047 – 0.0066 (+005.46%)	0.0139 – 0.1427 (+015.21%)	0.0066 – 0.0186 (+019.81%)		6
wbc					
Cnr1	0.1967 – 0.2078 (+005.63%)	0.9948 – 0.9762 (-001.87%)	0.5623 – 0.6167 (+009.68%)	+10	22
	0.0023 – 0.0051 (+002.31%)	0.0001 – 0.0114 (+001.14%)	0.0065 – 0.0128 (+001.56%)		1
Cnr2	0.1967 – 0.2060 (+004.74%)	0.9948 – 0.9710 (-002.40%)	0.5623 – 0.6182 (+009.96%)	+10	22
	0.0023 – 0.0054 (+002.70%)	0.0001 – 0.0123 (+001.24%)	0.0065 – 0.0151 (+002.93%)		1
Cnr3	0.1967 – 0.2073 (+005.39%)	0.9948 – 0.9755 (-001.94%)	0.5622 – 0.6164 (+009.65%)	+10	23
	0.0026 – 0.0036 (+001.10%)	0.0001 – 0.0127 (+001.27%)	0.0072 – 0.0161 (+003.17%)		1
Cnr4	0.1968 – 0.2064 (+004.87%)	0.9948 – 0.9813 (-001.36%)	0.5625 – 0.6062 (+007.78%)	+10	22
	0.0046 – 0.0048 (+001.66%)	0.0001 – 0.0099 (+001.00%)	0.0134 – 0.0148 (+002.23%)		2
Cnr5	0.1970 – 0.2050 (+004.06%)	0.9949 – 0.9726 (-002.24%)	0.5631 – 0.6132 (+008.90%)	+10	21
	0.0032 – 0.0053 (+001.98%)	0.0001 – 0.0105 (+001.05%)	0.0101 – 0.0202 (+003.39%)		0
Cnr6	0.1969 – 0.2073 (+005.29%)	0.9948 – 0.9718 (-002.32%)	0.5628 – 0.6208 (+010.32%)	+10	21
	0.0028 – 0.0031 (+001.84%)	0.0001 – 0.0051 (+000.52%)	0.0088 – 0.0115 (+002.20%)		0
Cnr7	0.1969 – 0.2076 (+005.44%)	0.9948 – 0.9808 (-001.41%)	0.5628 – 0.6103 (+008.45%)	+10	21
	0.0028 – 0.0025 (+001.28%)	0.0001 – 0.0040 (+000.40%)	0.0088 – 0.0059 (+001.18%)		0
Cnr8	0.1972 – 0.2058 (+004.39%)	0.9948 – 0.9750 (-002.00%)	0.5634 – 0.6120 (+008.65%)	+10	21
	0.0047 – 0.0058 (+002.54%)	0.0001 – 0.0087 (+000.87%)	0.0135 – 0.0094 (+001.62%)		0

Tabela A.6: Grupo 1 – Função de avaliação multi-objetivo – *Fnc1*

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
breast				
Cnr1	0.2245 – 0.1925 (-012.38%)	1.0000 – 0.9965 (-000.35%)	+8-2	46
	0.0360 – 0.0937 (+043.55%)	0.0000 – 0.0082 (+000.82%)		34
Cnr2	0.2245 – 0.3826 (+077.21%)	1.0000 – 0.8263 (-017.37%)	+9-1	66
	0.0245 – 0.1888 (+098.97%)	0.0000 – 0.1482 (+014.82%)		27

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr3	0.2255 – 0.2245 (+007.12%) 0.0644 – 0.0796 (+044.38%)	1.0000 – 0.9700 (-003.00%) 0.0000 – 0.0350 (+003.50%)	+8-2	34 24
Cnr4	0.2255 – 0.2469 (+012.58%) 0.0644 – 0.0921 (+035.07%)	1.0000 – 0.9350 (-006.50%) 0.0000 – 0.1068 (+010.68%)	+9-1	32 24
Cnr5	0.2245 – 0.1042 (-050.76%) 0.0340 – 0.0690 (+040.32%)	1.0000 – 0.9834 (-001.66%) 0.0000 – 0.0258 (+002.58%)	+6-4	26 6
Cnr6	0.2245 – 0.2100 (-004.01%) 0.0379 – 0.0912 (+044.99%)	1.0000 – 0.9655 (-003.45%) 0.0000 – 0.0591 (+005.91%)	+8-2	33 25
Cnr7	0.2245 – 0.2306 (+002.24%) 0.0204 – 0.0376 (+009.96%)	1.0000 – 0.9900 (-001.00%) 0.0000 – 0.0129 (+001.29%)	+9-1	21 2
Cnr8	0.2245 – 0.2245 (+000.00%) 0.0204 – 0.0204 (+000.00%)	1.0000 – 0.9875 (-001.25%) 0.0000 – 0.0177 (+001.77%)	+10	21 0
cmc				
Cnr1	0.0145 – 0.1072 (+638.82%) 0.0022 – 0.1158 (+830.72%)	1.0000 – 0.9762 (-002.37%) 0.0000 – 0.0296 (+002.96%)	+10	68 23
Cnr2	0.0126 – 0.1476 (+1134.74%) 0.0020 – 0.1950 (+1641.74%)	1.0000 – 0.9469 (-005.31%) 0.0000 – 0.0888 (+008.88%)	+9-1	49 30
Cnr3	0.0145 – 0.0222 (+056.88%) 0.0030 – 0.0177 (+133.58%)	1.0000 – 0.9965 (-000.35%) 0.0000 – 0.0048 (+000.48%)	+9-1	25 12
Cnr4	0.0144 – 0.0222 (+056.38%) 0.0025 – 0.0215 (+151.93%)	1.0000 – 0.9967 (-000.33%) 0.0000 – 0.0039 (+000.39%)	+9-1	24 6
Cnr5	0.0141 – 0.0333 (+165.36%) 0.0030 – 0.0453 (+392.90%)	1.0000 – 0.9961 (-000.38%) 0.0000 – 0.0034 (+000.34%)	+7-3	40 25
Cnr6	0.0152 – 0.0287 (+089.59%) 0.0026 – 0.0315 (+194.41%)	1.0000 – 0.9898 (-001.02%) 0.0000 – 0.0201 (+002.01%)	+8-2	22 2
Cnr7	0.0145 – 0.0150 (+003.80%) 0.0022 – 0.0083 (+056.12%)	1.0000 – 0.9976 (-000.24%) 0.0000 – 0.0029 (+000.29%)	+8-2	21 0
Cnr8	0.0145 – 0.0165 (+005.26%) 0.0030 – 0.0110 (+061.80%)	1.0000 – 0.9953 (-000.47%) 0.0000 – 0.0044 (+000.44%)	+7-3	21 0
heart				
Cnr1	0.2220 – 0.2780 (+023.67%) 0.0412 – 0.1474 (+055.27%)	1.0000 – 0.9703 (-002.97%) 0.0000 – 0.0489 (+004.89%)	+10	47 29
Cnr2	0.2333 – 0.4593 (+096.90%) 0.0408 – 0.1655 (+067.66%)	1.0000 – 0.9293 (-007.07%) 0.0000 – 0.0543 (+005.43%)	+9-1	57 26
Cnr3	0.2397 – 0.2207 (-006.04%) 0.0348 – 0.0582 (+027.05%)	1.0000 – 0.9797 (-002.03%) 0.0000 – 0.0141 (+001.41%)	+5-5	39 24
Cnr4	0.2476 – 0.3120 (+034.87%) 0.0502 – 0.1099 (+064.54%)	1.0000 – 0.9563 (-004.37%) 0.0000 – 0.0510 (+005.10%)	+7-3	70 32
Cnr5	0.2300 – 0.2640 (+015.36%) 0.0222 – 0.0704 (+030.43%)	1.0000 – 0.9727 (-002.73%) 0.0000 – 0.0240 (+002.40%)	+8-2	35 23
Cnr6	0.2333 – 0.2357 (+001.97%) 0.0408 – 0.0629 (+022.52%)	1.0000 – 0.9750 (-002.50%) 0.0000 – 0.0220 (+002.20%)	+5-5	27 9
Cnr7	0.2380 – 0.2730 (+018.19%) 0.0381 – 0.1067 (+047.94%)	1.0000 – 0.9713 (-002.87%) 0.0000 – 0.0278 (+002.78%)	+7-3	56 38
Cnr8	0.2476 – 0.1933 (-017.75%) 0.0502 – 0.0312 (+025.92%)	1.0000 – 0.9857 (-001.43%) 0.0000 – 0.0080 (+000.80%)	+4-6	21 1
nursery				
Cnr1	0.3333 – 1.0000 (+200.08%) 0.0060 – 0.0000 (+005.39%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr2	0.3333 – 1.0000 (+200.06%) 0.0047 – 0.0000 (+004.28%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	23 0
Cnr3	0.3333 – 1.0000 (+200.11%) 0.0066 – 0.0000 (+005.98%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	25 3
Cnr4	0.3333 – 1.0000 (+200.18%) 0.0088 – 0.0000 (+007.97%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	23 2

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr5	0.3333 – 1.0000 (+200.06%) 0.0047 – 0.0000 (+004.28%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr6	0.3333 – 1.0000 (+200.18%) 0.0088 – 0.0000 (+007.97%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr7	0.3333 – 1.0000 (+200.21%) 0.0097 – 0.0000 (+008.75%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	21 0
Cnr8	0.3333 – 1.0000 (+200.11%) 0.0066 – 0.0000 (+005.98%)	1.0000 – 1.0000 (+000.00%) 0.0000 – 0.0000 (+000.00%)	+10	24 1
spambase				
Cnr1	0.2417 – 0.3307 (+037.10%) 0.0121 – 0.1984 (+083.03%)	1.0000 – 0.9739 (-002.60%) 0.0000 – 0.0804 (+008.04%)	+10	23 6
Cnr2	0.2417 – 0.2674 (+010.66%) 0.0121 – 0.0101 (+002.18%)	1.0000 – 0.9996 (-000.04%) 0.0000 – 0.0006 (+000.06%)	+10	21 0
Cnr3	0.2417 – 0.2671 (+010.54%) 0.0121 – 0.0119 (+001.58%)	1.0000 – 0.9997 (-000.03%) 0.0000 – 0.0005 (+000.05%)	+10	21 0
Cnr4	0.2418 – 0.5194 (+114.93%) 0.0088 – 0.3299 (+137.24%)	1.0000 – 0.8871 (-011.29%) 0.0000 – 0.1692 (+016.92%)	+10	38 31
Cnr5	0.2417 – 0.5605 (+132.40%) 0.0077 – 0.3618 (+150.48%)	1.0000 – 0.8541 (-014.58%) 0.0000 – 0.1593 (+015.93%)	+10	41 24
Cnr6	0.2417 – 0.3394 (+040.15%) 0.0077 – 0.2309 (+093.95%)	1.0000 – 0.9208 (-007.92%) 0.0000 – 0.2461 (+024.61%)	+10	25 13
Cnr7	0.2417 – 0.2691 (+011.37%) 0.0077 – 0.0043 (+002.52%)	1.0000 – 0.9993 (-000.07%) 0.0000 – 0.0008 (+000.08%)	+10	22 5
Cnr8	0.2418 – 0.6636 (+174.04%) 0.0084 – 0.3436 (+141.26%)	1.0000 – 0.7715 (-022.85%) 0.0000 – 0.2213 (+022.13%)	+10	30 11
vehicle				
Cnr1	0.2355 – 0.6784 (+290.38%) 0.1627 – 0.3579 (+272.13%)	1.0000 – 0.7503 (-024.97%) 0.0000 – 0.2296 (+022.96%)	+9-1	34 13
Cnr2	0.2717 – 0.6235 (+196.37%) 0.1747 – 0.1958 (+157.38%)	1.0000 – 0.7830 (-021.70%) 0.0000 – 0.2008 (+020.08%)	+9-1	52 22
Cnr3	0.3026 – 0.5717 (+145.74%) 0.1811 – 0.3218 (+173.51%)	1.0000 – 0.8708 (-012.92%) 0.0000 – 0.1738 (+017.38%)	+8-2	41 22
Cnr4	0.2537 – 0.3910 (+087.06%) 0.1871 – 0.2486 (+106.24%)	1.0000 – 0.9391 (-006.09%) 0.0000 – 0.0989 (+009.89%)	+9-1	46 26
Cnr5	0.2355 – 0.5637 (+231.18%) 0.1627 – 0.3749 (+213.75%)	1.0000 – 0.8075 (-019.25%) 0.0000 – 0.2838 (+028.38%)	+9-1	33 11
Cnr6	0.1946 – 0.2668 (+077.08%) 0.1151 – 0.1796 (+157.27%)	1.0000 – 0.9916 (-000.84%) 0.0000 – 0.0049 (+000.49%)	+8-2	27 6
Cnr7	0.1946 – 0.8903 (+440.83%) 0.1151 – 0.1563 (+207.36%)	1.0000 – 0.6587 (-034.13%) 0.0000 – 0.2170 (+021.70%)	+10	36 8
Cnr8	0.2355 – 0.3043 (+076.64%) 0.1627 – 0.1734 (+132.69%)	1.0000 – 0.9959 (-000.41%) 0.0000 – 0.0021 (+000.21%)	+7-3	25 6
wbc				
Cnr1	0.8649 – 0.9189 (+006.24%) 0.0099 – 0.0493 (+005.39%)	1.0000 – 0.9691 (-003.09%) 0.0000 – 0.0350 (+003.50%)	+10	26 8
Cnr2	0.8649 – 0.9099 (+005.20%) 0.0092 – 0.0388 (+004.31%)	1.0000 – 0.9791 (-002.09%) 0.0000 – 0.0164 (+001.64%)	+10	25 7
Cnr3	0.8649 – 0.9117 (+005.41%) 0.0188 – 0.0312 (+002.56%)	1.0000 – 0.9875 (-001.25%) 0.0000 – 0.0168 (+001.68%)	+10	32 24
Cnr4	0.8649 – 0.9135 (+005.62%) 0.0102 – 0.0448 (+004.97%)	1.0000 – 0.9666 (-003.34%) 0.0000 – 0.0338 (+003.38%)	+10	42 25
Cnr5	0.8667 – 0.9004 (+003.90%) 0.0206 – 0.0419 (+004.15%)	1.0000 – 0.9765 (-002.35%) 0.0000 – 0.0246 (+002.46%)	+10	28 15
Cnr6	0.8658 – 0.9275 (+007.16%) 0.0129 – 0.0333 (+004.42%)	1.0000 – 0.9640 (-003.60%) 0.0000 – 0.0228 (+002.28%)	+10	25 2

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr7	0.8662 – 0.8856 (+002.24%)	1.0000 – 0.9975 (-000.25%)	+10	21
	0.0147 – 0.0350 (+003.73%)	0.0000 – 0.0040 (+000.40%)		0
Cnr8	0.8667 – 0.8851 (+002.13%)	1.0000 – 0.9857 (-001.43%)	+10	25
	0.0206 – 0.0368 (+003.46%)	0.0000 – 0.0221 (+002.21%)		12

Tabela A.7: Grupo 1 – Função de avaliação multi-objetivo – *Fnc2*

A.2 Grupo 2

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	Laplace Inicial – Final (Variação%)	Suporte Inicial – Final (Variação%)	+/-	#Iterações
breast					
Cnr1	0.0325 – 0.0700 (+155.97%)	0.9093 – 0.7795 (-014.00%)	0.0964 – 0.4834 (+510.27%)	+10	27
	0.0139 – 0.0137 (+138.97%)	0.0439 – 0.1153 (+013.94%)	0.0424 – 0.1152 (+356.88%)		5
Cnr2	0.0617 – 0.0667 (+029.97%)	0.8880 – 0.8122 (-008.31%)	0.2754 – 0.5123 (+186.89%)	+8-2	24
	0.0226 – 0.0128 (+086.75%)	0.0410 – 0.0314 (+006.27%)	0.1592 – 0.0583 (+232.13%)		6
Cnr3	0.0730 – 0.0767 (+012.40%)	0.8929 – 0.8452 (-005.03%)	0.3275 – 0.4689 (+099.93%)	+8-2	25
	0.0220 – 0.0142 (+032.68%)	0.0508 – 0.0351 (+007.21%)	0.1634 – 0.0768 (+149.17%)		4
Cnr4	0.0451 – 0.0672 (+113.62%)	0.9220 – 0.8177 (-011.26%)	0.1841 – 0.4058 (+229.39%)	+8-2	27
	0.0293 – 0.0120 (+151.65%)	0.0390 – 0.0676 (+007.26%)	0.1533 – 0.1408 (+211.91%)		7
Cnr5	0.0378 – 0.0621 (+098.51%)	0.9256 – 0.8180 (-011.45%)	0.1290 – 0.4449 (+352.85%)	+8-2	26
	0.0131 – 0.0182 (+120.18%)	0.0375 – 0.0316 (+005.66%)	0.0537 – 0.0680 (+310.77%)		3
Cnr6	0.0430 – 0.0632 (+081.32%)	0.9157 – 0.7898 (-013.61%)	0.1855 – 0.4225 (+198.25%)	+8-2	24
	0.0204 – 0.0205 (+124.18%)	0.0475 – 0.0756 (+008.69%)	0.1405 – 0.1173 (+168.12%)		2
Cnr7	0.0473 – 0.0690 (+137.61%)	0.8815 – 0.8352 (-005.10%)	0.1609 – 0.4565 (+622.80%)	+7-3	25
	0.0301 – 0.0166 (+176.97%)	0.0477 – 0.0390 (+005.14%)	0.1683 – 0.1124 (+734.31%)		3
Cnr8	0.0635 – 0.0542 (-003.72%)	0.9209 – 0.8047 (-012.54%)	0.2500 – 0.3326 (+083.43%)	+5-5	27
	0.0281 – 0.0176 (+037.56%)	0.0399 – 0.1334 (+014.76%)	0.1689 – 0.1325 (+131.82%)		7
cmc					
Cnr1	0.0110 – 0.0449 (+421.18%)	0.7977 – 0.4184 (-046.81%)	0.0202 – 0.2024 (+1487.62%)	+10	28
	0.0081 – 0.0168 (+298.67%)	0.0800 – 0.0467 (+009.76%)	0.0152 – 0.0870 (+1533.47%)		7
Cnr2	0.0138 – 0.0526 (+377.53%)	0.7982 – 0.4395 (-044.35%)	0.0246 – 0.1431 (+648.94%)	+10	22
	0.0093 – 0.0125 (+212.47%)	0.0980 – 0.0947 (+012.86%)	0.0167 – 0.0388 (+388.76%)		1
Cnr3	0.0146 – 0.0500 (+491.68%)	0.7965 – 0.5645 (-028.61%)	0.0261 – 0.1424 (+820.23%)	+10	36
	0.0121 – 0.0094 (+529.71%)	0.1132 – 0.1040 (+013.90%)	0.0214 – 0.0247 (+712.45%)		15
Cnr4	0.0109 – 0.0484 (+650.40%)	0.7796 – 0.4372 (-043.21%)	0.0201 – 0.1570 (+978.51%)	+9-1	29
	0.0093 – 0.0145 (+569.62%)	0.0891 – 0.1115 (+016.07%)	0.0162 – 0.0805 (+685.92%)		10
Cnr5	0.0085 – 0.0520 (+567.45%)	0.8058 – 0.4245 (-046.99%)	0.0151 – 0.1500 (+985.68%)	+10	25
	0.0023 – 0.0103 (+248.24%)	0.0549 – 0.0635 (+009.66%)	0.0048 – 0.0586 (+479.77%)		4
Cnr6	0.0080 – 0.0528 (+665.56%)	0.7586 – 0.3793 (-049.61%)	0.0156 – 0.1545 (+1235.03%)	+10	22
	0.0037 – 0.0101 (+308.42%)	0.0738 – 0.0421 (+007.04%)	0.0084 – 0.0424 (+897.92%)		2
Cnr7	0.0077 – 0.0461 (+570.09%)	0.7988 – 0.4190 (-047.12%)	0.0132 – 0.1842 (+1364.64%)	+9-1	39
	0.0026 – 0.0196 (+341.63%)	0.0751 – 0.0754 (+010.58%)	0.0051 – 0.0990 (+735.53%)		32
Cnr8	0.0197 – 0.0486 (+391.62%)	0.8483 – 0.3720 (-055.84%)	0.0342 – 0.1476 (+764.92%)	+9-1	21
	0.0145 – 0.0122 (+402.97%)	0.0877 – 0.0403 (+005.51%)	0.0265 – 0.0356 (+707.07%)		0
heart					
Cnr1	0.0951 – 0.1297 (+061.97%)	0.9290 – 0.8280 (-010.82%)	0.2118 – 0.3622 (+108.28%)	+8-2	23
	0.0391 – 0.0068 (+069.44%)	0.0205 – 0.0665 (+007.61%)	0.0957 – 0.0434 (+095.72%)		2
Cnr2	0.0676 – 0.1230 (+100.55%)	0.9366 – 0.8366 (-010.72%)	0.1422 – 0.3437 (+179.65%)	+9-1	24
	0.0288 – 0.0068 (+053.60%)	0.0146 – 0.0502 (+004.48%)	0.0782 – 0.0278 (+092.11%)		4
Cnr3	0.0797 – 0.1243 (+074.25%)	0.9400 – 0.8165 (-013.17%)	0.1748 – 0.3556 (+135.13%)	+8-2	28
	0.0333 – 0.0171 (+055.09%)	0.0185 – 0.0656 (+006.41%)	0.0972 – 0.0500 (+066.57%)		7

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	Laplace Inicial – Final (Variação%)	Suporte Inicial – Final (Variação%)	+/-	#Iterações
Cnr4	0.0709 – 0.1136 (+069.67%)	0.9464 – 0.7571 (-020.02%)	0.1489 – 0.3341 (+131.77%)	+9-1	24
	0.0190 – 0.0166 (+047.48%)	0.0134 – 0.0802 (+008.29%)	0.0309 – 0.0769 (+063.35%)		3
Cnr5	0.0632 – 0.1288 (+115.41%)	0.9384 – 0.7999 (-014.82%)	0.1296 – 0.3792 (+207.67%)	+10	22
	0.0218 – 0.0210 (+046.58%)	0.0160 – 0.0829 (+008.05%)	0.0365 – 0.0527 (+074.69%)		1
Cnr6	0.1064 – 0.1219 (+030.69%)	0.9241 – 0.8263 (-010.35%)	0.2607 – 0.3578 (+066.26%)	+6-4	22
	0.0424 – 0.0168 (+049.47%)	0.0310 – 0.0597 (+008.94%)	0.1112 – 0.0679 (+087.46%)		1
Cnr7	0.0866 – 0.1268 (+065.05%)	0.9440 – 0.8252 (-012.56%)	0.1926 – 0.3570 (+115.39%)	+8-2	24
	0.0382 – 0.0124 (+049.30%)	0.0145 – 0.0876 (+009.55%)	0.0943 – 0.0534 (+072.02%)		5
Cnr8	0.0953 – 0.1282 (+054.62%)	0.9218 – 0.8492 (-007.99%)	0.2044 – 0.3393 (+095.65%)	+9-1	27
	0.0407 – 0.0183 (+056.95%)	0.0393 – 0.0901 (+007.85%)	0.0994 – 0.0476 (+071.40%)		9
nursery					
Cnr1	0.0688 – 0.2222 (+346.86%)	0.9919 – 0.9982 (+000.64%)	0.1031 – 0.3333 (+346.67%)	+10	23
	0.0567 – 0.0000 (+213.04%)	0.0039 – 0.0000 (+000.39%)	0.0850 – 0.0000 (+212.94%)		2
Cnr2	0.0840 – 0.2222 (+336.05%)	0.9920 – 0.9982 (+000.62%)	0.1260 – 0.3333 (+336.16%)	+10	23
	0.0748 – 0.0000 (+261.93%)	0.0048 – 0.0000 (+000.48%)	0.1122 – 0.0000 (+262.07%)		2
Cnr3	0.0527 – 0.2222 (+534.84%)	0.9884 – 0.9982 (+000.99%)	0.0790 – 0.3333 (+534.75%)	+10	23
	0.0603 – 0.0000 (+235.44%)	0.0043 – 0.0000 (+000.43%)	0.0904 – 0.0000 (+235.39%)		2
Cnr4	0.0800 – 0.2222 (+407.23%)	0.9908 – 0.9982 (+000.75%)	0.1200 – 0.3333 (+407.62%)	+10	24
	0.0770 – 0.0000 (+331.32%)	0.0060 – 0.0000 (+000.61%)	0.1155 – 0.0000 (+331.68%)		2
Cnr5	0.0615 – 0.2222 (+449.51%)	0.9900 – 0.9982 (+000.83%)	0.0923 – 0.3333 (+449.51%)	+10	22
	0.0594 – 0.0000 (+274.47%)	0.0050 – 0.0000 (+000.51%)	0.0891 – 0.0000 (+274.42%)		1
Cnr6	0.0624 – 0.2222 (+445.85%)	0.9894 – 0.9982 (+000.89%)	0.0933 – 0.3333 (+447.31%)	+10	22
	0.0590 – 0.0000 (+307.18%)	0.0058 – 0.0000 (+000.59%)	0.0885 – 0.0000 (+306.85%)		1
Cnr7	0.0715 – 0.2084 (+348.67%)	0.9917 – 0.9551 (-003.66%)	0.1071 – 0.3186 (+350.94%)	+9-1	23
	0.0587 – 0.0437 (+254.09%)	0.0044 – 0.1363 (+013.89%)	0.0881 – 0.0465 (+251.32%)		1
Cnr8	0.0514 – 0.2222 (+400.01%)	0.9909 – 0.9982 (+000.74%)	0.0772 – 0.3333 (+400.07%)	+10	23
	0.0192 – 0.0000 (+213.07%)	0.0039 – 0.0000 (+000.39%)	0.0288 – 0.0000 (+213.20%)		2
spambase					
Cnr1	0.0823 – 0.1589 (+104.54%)	0.9737 – 0.8822 (-009.36%)	0.2001 – 0.5058 (+176.52%)	+10	26
	0.0204 – 0.0263 (+065.91%)	0.0156 – 0.0295 (+003.64%)	0.0540 – 0.0638 (+113.82%)		7
Cnr2	0.0970 – 0.1588 (+072.67%)	0.9447 – 0.8783 (-006.53%)	0.2361 – 0.4858 (+126.76%)	+10	29
	0.0215 – 0.0133 (+048.01%)	0.0623 – 0.0389 (+009.21%)	0.1091 – 0.0816 (+067.49%)		13
Cnr3	0.0963 – 0.1541 (+078.58%)	0.9418 – 0.8325 (-011.27%)	0.2133 – 0.4916 (+169.00%)	+10	28
	0.0290 – 0.0196 (+073.06%)	0.0647 – 0.0682 (+008.80%)	0.1104 – 0.0877 (+119.10%)		4
Cnr4	0.0938 – 0.1598 (+074.36%)	0.9664 – 0.8758 (-009.34%)	0.1908 – 0.4989 (+173.40%)	+10	27
	0.0156 – 0.0192 (+032.52%)	0.0204 – 0.0360 (+004.14%)	0.0406 – 0.0944 (+081.24%)		3
Cnr5	0.0889 – 0.1742 (+102.26%)	0.9521 – 0.8994 (-005.06%)	0.2269 – 0.5056 (+145.34%)	+10	24
	0.0158 – 0.0055 (+041.21%)	0.0633 – 0.0304 (+008.43%)	0.0987 – 0.0681 (+069.69%)		2
Cnr6	0.0830 – 0.1643 (+112.44%)	0.9719 – 0.8801 (-009.43%)	0.1695 – 0.4986 (+233.55%)	+10	24
	0.0251 – 0.0119 (+057.00%)	0.0197 – 0.0363 (+003.72%)	0.0563 – 0.0666 (+137.00%)		2
Cnr7	0.0912 – 0.1585 (+085.17%)	0.9684 – 0.8716 (-009.94%)	0.1888 – 0.5238 (+228.65%)	+10	25
	0.0228 – 0.0202 (+055.20%)	0.0147 – 0.0484 (+005.79%)	0.0593 – 0.0263 (+197.63%)		4
Cnr8	0.0970 – 0.1580 (+071.60%)	0.9643 – 0.8770 (-008.98%)	0.2023 – 0.5129 (+160.92%)	+10	25
	0.0229 – 0.0131 (+043.13%)	0.0220 – 0.0360 (+004.88%)	0.0356 – 0.0232 (+048.62%)		3
vehicle					
Cnr1	0.0534 – 0.1242 (+205.49%)	0.8914 – 0.5554 (-037.74%)	0.0716 – 0.2246 (+324.77%)	+10	30
	0.0256 – 0.0266 (+198.90%)	0.0514 – 0.0943 (+009.85%)	0.0342 – 0.0353 (+277.67%)		6
Cnr2	0.0679 – 0.1161 (+123.97%)	0.9075 – 0.5953 (-034.30%)	0.0901 – 0.2045 (+194.67%)	+9-1	33
	0.0296 – 0.0184 (+162.54%)	0.0420 – 0.1454 (+016.13%)	0.0387 – 0.0402 (+228.03%)		8
Cnr3	0.0592 – 0.1175 (+122.64%)	0.9075 – 0.6902 (-024.31%)	0.0790 – 0.1905 (+185.11%)	+10	32
	0.0247 – 0.0259 (+077.82%)	0.0377 – 0.1637 (+016.17%)	0.0331 – 0.0450 (+143.08%)		10
Cnr4	0.0661 – 0.1027 (+075.51%)	0.9145 – 0.6771 (-025.95%)	0.0880 – 0.1714 (+120.40%)	+10	33
	0.0260 – 0.0133 (+064.97%)	0.0303 – 0.1840 (+019.91%)	0.0338 – 0.0428 (+092.58%)		9
Cnr5	0.0586 – 0.1118 (+157.31%)	0.8899 – 0.6267 (-029.70%)	0.0785 – 0.1903 (+233.93%)	+10	25
	0.0286 – 0.0252 (+190.26%)	0.0585 – 0.1363 (+013.66%)	0.0380 – 0.0433 (+277.05%)		4

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Novidade Inicial – Final (Variação%)	Laplace Inicial – Final (Variação%)	Suporte Inicial – Final (Variação%)	+/-	#Iterações
Cnr6	0.0697 – 0.1077 (+070.39%)	0.9156 – 0.5722 (-037.62%)	0.0936 – 0.1920 (+136.02%)	+9-1	27
	0.0235 – 0.0308 (+072.15%)	0.0388 – 0.1417 (+014.63%)	0.0316 – 0.0381 (+115.97%)		6
Cnr7	0.0609 – 0.1162 (+098.04%)	0.9127 – 0.5816 (-036.38%)	0.0816 – 0.2064 (+169.29%)	+10	30
	0.0203 – 0.0316 (+041.79%)	0.0308 – 0.0984 (+009.95%)	0.0269 – 0.0458 (+068.21%)		10
Cnr8	0.0708 – 0.0994 (+055.59%)	0.9185 – 0.6199 (-032.62%)	0.0943 – 0.1730 (+111.10%)	+9-1	27
	0.0233 – 0.0184 (+062.95%)	0.0293 – 0.1710 (+018.22%)	0.0305 – 0.0438 (+122.95%)		6
wbc					
Cnr1	0.1967 – 0.2053 (+004.37%)	0.9948 – 0.9691 (-002.59%)	0.5622 – 0.6181 (+009.98%)	+10	22
	0.0046 – 0.0054 (+002.37%)	0.0001 – 0.0078 (+000.78%)	0.0128 – 0.0130 (+002.44%)		1
Cnr2	0.1967 – 0.2017 (+002.52%)	0.9948 – 0.9771 (-001.78%)	0.5622 – 0.5977 (+006.26%)	+8-2	21
	0.0040 – 0.0067 (+002.20%)	0.0001 – 0.0080 (+000.81%)	0.0119 – 0.0299 (+003.82%)		0
Cnr3	0.1967 – 0.2072 (+005.38%)	0.9948 – 0.9746 (-002.04%)	0.5622 – 0.6173 (+009.79%)	+10	22
	0.0040 – 0.0022 (+001.93%)	0.0001 – 0.0094 (+000.95%)	0.0119 – 0.0145 (+001.19%)		1
Cnr4	0.1967 – 0.2070 (+005.23%)	0.9948 – 0.9791 (-001.58%)	0.5622 – 0.6105 (+008.61%)	+10	22
	0.0040 – 0.0034 (+001.33%)	0.0001 – 0.0057 (+000.58%)	0.0119 – 0.0135 (+001.64%)		1
Cnr5	0.1967 – 0.2054 (+004.42%)	0.9948 – 0.9713 (-002.37%)	0.5622 – 0.6158 (+009.53%)	+10	21
	0.0040 – 0.0036 (+001.67%)	0.0001 – 0.0068 (+000.69%)	0.0119 – 0.0174 (+001.30%)		0
Cnr6	0.1968 – 0.2059 (+004.67%)	0.9948 – 0.9768 (-001.81%)	0.5622 – 0.6102 (+008.55%)	+10	21
	0.0033 – 0.0033 (+001.50%)	0.0001 – 0.0068 (+000.68%)	0.0079 – 0.0101 (+001.62%)		0
Cnr7	0.1968 – 0.2049 (+004.16%)	0.9948 – 0.9773 (-001.77%)	0.5622 – 0.6067 (+007.92%)	+10	23
	0.0033 – 0.0031 (+002.20%)	0.0001 – 0.0093 (+000.93%)	0.0079 – 0.0125 (+001.98%)		4
Cnr8	0.1968 – 0.2046 (+003.99%)	0.9948 – 0.9779 (-001.70%)	0.5622 – 0.6052 (+007.63%)	+10	21
	0.0033 – 0.0035 (+001.75%)	0.0001 – 0.0106 (+001.07%)	0.0079 – 0.0184 (+002.48%)		0

Tabela A.8: Grupo 2 – Função de avaliação multi-objetivo – *Fnc1*

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
breast				
Cnr1	0.1876 – 0.2340 (+054.97%)	1.0000 – 0.9499 (-005.01%)	+7-3	53
	0.0727 – 0.0984 (+118.81%)	0.0000 – 0.0612 (+006.12%)		32
Cnr2	0.1716 – 0.3392 (+120.23%)	1.0000 – 0.8882 (-011.18%)	+10	50
	0.0748 – 0.1146 (+089.93%)	0.0000 – 0.0993 (+009.93%)		29
Cnr3	0.1590 – 0.1968 (+050.43%)	1.0000 – 0.9600 (-004.00%)	+10	53
	0.0491 – 0.1228 (+148.20%)	0.0000 – 0.0603 (+006.03%)		34
Cnr4	0.1362 – 0.1942 (+046.26%)	1.0000 – 0.9658 (-003.42%)	+9-1	67
	0.0456 – 0.0948 (+062.37%)	0.0000 – 0.0392 (+003.92%)		33
Cnr5	0.1441 – 0.1545 (+013.11%)	1.0000 – 0.9869 (-001.31%)	+9-1	28
	0.0460 – 0.0767 (+053.89%)	0.0000 – 0.0139 (+001.39%)		7
Cnr6	0.1404 – 0.1737 (+032.94%)	1.0000 – 0.9399 (-006.01%)	+9-1	33
	0.0479 – 0.1338 (+113.53%)	0.0000 – 0.0966 (+009.66%)		16
Cnr7	0.1683 – 0.1683 (-001.27%)	1.0000 – 0.9705 (-002.95%)	+8-2	24
	0.0585 – 0.0852 (+049.64%)	0.0000 – 0.0351 (+003.51%)		6
Cnr8	0.1654 – 0.1502 (-010.02%)	1.0000 – 0.9615 (-003.85%)	+8-2	23
	0.0482 – 0.0888 (+049.19%)	0.0000 – 0.0549 (+005.49%)		5
cmc				
Cnr1	0.0290 – 0.0953 (+343.46%)	1.0000 – 0.9811 (-001.89%)	+10	51
	0.0107 – 0.1235 (+704.51%)	0.0000 – 0.0415 (+004.15%)		36
Cnr2	0.0213 – 0.1249 (+609.95%)	1.0000 – 0.9552 (-004.48%)	+10	60
	0.0089 – 0.1076 (+716.21%)	0.0000 – 0.0480 (+004.80%)		30
Cnr3	0.0226 – 0.0723 (+244.73%)	1.0000 – 0.9951 (-000.49%)	+8-2	43
	0.0065 – 0.0518 (+263.95%)	0.0000 – 0.0073 (+000.73%)		23
Cnr4	0.0415 – 0.0656 (+102.82%)	1.0000 – 0.9932 (-000.67%)	+8-2	46
	0.0457 – 0.0593 (+187.34%)	0.0000 – 0.0108 (+001.08%)		33
Cnr5	0.0259 – 0.0371 (+059.11%)	1.0000 – 0.9939 (-000.61%)	+8-2	40
	0.0109 – 0.0233 (+098.12%)	0.0000 – 0.0052 (+000.52%)		22

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr6	0.0427 – 0.0377 (+014.93%)	1.0000 – 0.9947 (-000.53%)	+8-2	33
	0.0451 – 0.0398 (+101.78%)	0.0000 – 0.0109 (+001.09%)		24
Cnr7	0.0230 – 0.0482 (+094.10%)	1.0000 – 0.9956 (-000.44%)	+10	36
	0.0079 – 0.0394 (+112.67%)	0.0000 – 0.0046 (+000.46%)		30
Cnr8	0.0286 – 0.0314 (+008.59%)	1.0000 – 0.9975 (-000.25%)	+8-2	22
	0.0080 – 0.0187 (+062.74%)	0.0000 – 0.0031 (+000.31%)		2
heart				
Cnr1	0.2407 – 0.2830 (+016.79%)	1.0000 – 0.9773 (-002.27%)	+9-1	48
	0.0266 – 0.1220 (+049.95%)	0.0000 – 0.0202 (+002.02%)		24
Cnr2	0.2377 – 0.4240 (+083.04%)	1.0000 – 0.9273 (-007.27%)	+9-1	43
	0.0509 – 0.1860 (+090.45%)	0.0000 – 0.0745 (+007.45%)		24
Cnr3	0.2343 – 0.2910 (+023.55%)	1.0000 – 0.9720 (-002.80%)	+9-1	28
	0.0344 – 0.1084 (+043.07%)	0.0000 – 0.0218 (+002.18%)		8
Cnr4	0.2513 – 0.2683 (+009.20%)	1.0000 – 0.9833 (-001.67%)	+7-3	58
	0.0307 – 0.1613 (+067.56%)	0.0000 – 0.0196 (+001.96%)		34
Cnr5	0.2487 – 0.2687 (+008.24%)	1.0000 – 0.9790 (-002.10%)	+7-3	30
	0.0108 – 0.1024 (+040.97%)	0.0000 – 0.0256 (+002.56%)		9
Cnr6	0.2503 – 0.2507 (+000.26%)	1.0000 – 0.9777 (-002.23%)	+6-4	51
	0.0226 – 0.1632 (+064.87%)	0.0000 – 0.0266 (+002.66%)		33
Cnr7	0.2423 – 0.2443 (+000.12%)	1.0000 – 0.9763 (-002.37%)	+8-2	30
	0.0342 – 0.0747 (+021.82%)	0.0000 – 0.0321 (+003.21%)		22
Cnr8	0.2327 – 0.1970 (-014.00%)	1.0000 – 0.9890 (-001.10%)	+7-3	28
	0.0345 – 0.0565 (+024.69%)	0.0000 – 0.0135 (+001.35%)		10
nursery				
Cnr1	0.2533 – 0.9156 (+459.92%)	1.0000 – 1.0000 (+000.00%)	+10	27
	0.2705 – 0.2670 (+325.90%)	0.0000 – 0.0000 (+000.00%)		4
Cnr2	0.1641 – 1.0000 (+629.32%)	1.0000 – 1.0000 (+000.00%)	+10	26
	0.0911 – 0.0000 (+245.55%)	0.0000 – 0.0000 (+000.00%)		5
Cnr3	0.2515 – 1.0000 (+522.67%)	1.0000 – 1.0000 (+000.00%)	+10	26
	0.2696 – 0.0000 (+318.17%)	0.0000 – 0.0000 (+000.00%)		3
Cnr4	0.3053 – 1.0000 (+350.26%)	1.0000 – 1.0000 (+000.00%)	+10	26
	0.2542 – 0.0000 (+210.99%)	0.0000 – 0.0000 (+000.00%)		4
Cnr5	0.1982 – 1.0000 (+459.47%)	1.0000 – 1.0000 (+000.00%)	+10	22
	0.0770 – 0.0000 (+164.69%)	0.0000 – 0.0000 (+000.00%)		1
Cnr6	0.4124 – 1.0000 (+274.39%)	1.0000 – 1.0000 (+000.00%)	+10	22
	0.3274 – 0.0000 (+215.86%)	0.0000 – 0.0000 (+000.00%)		1
Cnr7	0.3639 – 1.0000 (+350.59%)	1.0000 – 1.0000 (+000.00%)	+10	23
	0.3407 – 0.0000 (+244.85%)	0.0000 – 0.0000 (+000.00%)		1
Cnr8	0.3052 – 1.0000 (+381.84%)	1.0000 – 1.0000 (+000.00%)	+10	24
	0.2694 – 0.0000 (+233.67%)	0.0000 – 0.0000 (+000.00%)		2
spambase				
Cnr1	0.1001 – 0.3103 (+416.45%)	1.0000 – 0.9097 (-009.03%)	+10	38
	0.0797 – 0.2490 (+715.10%)	0.0000 – 0.2821 (+028.21%)		16
Cnr2	0.1531 – 0.2831 (+225.45%)	1.0000 – 0.9961 (-000.39%)	+10	27
	0.0974 – 0.0412 (+299.75%)	0.0000 – 0.0062 (+000.62%)		12
Cnr3	0.0994 – 0.3080 (+325.61%)	1.0000 – 0.9032 (-009.68%)	+10	56
	0.0793 – 0.2676 (+455.18%)	0.0000 – 0.2952 (+029.52%)		34
Cnr4	0.1398 – 0.3581 (+279.81%)	1.0000 – 0.9061 (-009.39%)	+10	39
	0.0825 – 0.3010 (+435.57%)	0.0000 – 0.1836 (+018.36%)		20
Cnr5	0.0868 – 0.1981 (+097.48%)	1.0000 – 0.9545 (-004.55%)	+10	45
	0.0597 – 0.2665 (+103.60%)	0.0000 – 0.1395 (+013.95%)		32
Cnr6	0.1258 – 0.3077 (+374.58%)	1.0000 – 0.9137 (-008.63%)	+9-1	28
	0.0895 – 0.2616 (+764.29%)	0.0000 – 0.2489 (+024.89%)		6
Cnr7	0.1446 – 0.2314 (+072.77%)	1.0000 – 0.9598 (-004.02%)	+8-2	38
	0.0878 – 0.2611 (+124.22%)	0.0000 – 0.1230 (+012.30%)		28

continua na próxima página...

...continuação da página anterior

Conj. Dados Cenário	Sensibilidade Inicial – Final (Variação%)	Especificidade Inicial – Final (Variação%)	+/-	#Iterações
Cnr8	0.1073 – 0.2637 (+175.14%) 0.0775 – 0.2325 (+217.65%)	1.0000 – 0.9617 (-003.82%) 0.0000 – 0.1138 (+011.38%)	+9-1	29 7
vehicle				
Cnr1	0.3094 – 0.7239 (+222.05%) 0.1390 – 0.1374 (+254.66%)	1.0000 – 0.7650 (-023.50%) 0.0000 – 0.2202 (+022.02%)	+10	54 24
Cnr2	0.3160 – 0.7676 (+189.48%) 0.1082 – 0.1431 (+178.83%)	1.0000 – 0.8270 (-017.30%) 0.0000 – 0.1207 (+012.07%)	+10	60 22
Cnr3	0.3667 – 0.8195 (+143.88%) 0.1194 – 0.1885 (+092.37%)	1.0000 – 0.7165 (-028.35%) 0.0000 – 0.2911 (+029.11%)	+10	58 26
Cnr4	0.2964 – 0.4830 (+061.32%) 0.0782 – 0.2369 (+059.97%)	1.0000 – 0.9348 (-006.52%) 0.0000 – 0.1212 (+012.12%)	+9-1	53 28
Cnr5	0.2842 – 0.4805 (+080.53%) 0.1240 – 0.2930 (+149.27%)	1.0000 – 0.8516 (-014.84%) 0.0000 – 0.2907 (+029.07%)	+9-1	35 10
Cnr6	0.3374 – 0.3844 (+019.14%) 0.0755 – 0.3442 (+109.27%)	1.0000 – 0.9263 (-007.37%) 0.0000 – 0.1373 (+013.73%)	+8-2	29 7
Cnr7	0.3779 – 0.6890 (+114.28%) 0.1071 – 0.2087 (+133.80%)	1.0000 – 0.8098 (-019.02%) 0.0000 – 0.2666 (+026.66%)	+9-1	51 22
Cnr8	0.3180 – 0.3680 (+018.70%) 0.1267 – 0.2138 (+055.11%)	1.0000 – 0.9688 (-003.12%) 0.0000 – 0.0911 (+009.11%)	+10	36 24
wbc				
Cnr1	0.8649 – 0.9487 (+009.68%) 0.0200 – 0.0361 (+003.04%)	1.0000 – 0.9347 (-006.53%) 0.0000 – 0.0472 (+004.72%)	+10	42 18
Cnr2	0.8649 – 0.9131 (+005.56%) 0.0180 – 0.0472 (+004.89%)	1.0000 – 0.9473 (-005.27%) 0.0000 – 0.0510 (+005.10%)	+9-1	34 13
Cnr3	0.8649 – 0.9257 (+007.10%) 0.0180 – 0.0335 (+005.25%)	1.0000 – 0.9674 (-003.26%) 0.0000 – 0.0249 (+002.49%)	+10	35 16
Cnr4	0.8649 – 0.8919 (+003.14%) 0.0180 – 0.0300 (+003.37%)	1.0000 – 0.9841 (-001.59%) 0.0000 – 0.0175 (+001.75%)	+10	34 18
Cnr5	0.8649 – 0.8519 (-001.53%) 0.0180 – 0.0802 (+008.40%)	1.0000 – 0.9848 (-001.52%) 0.0000 – 0.0144 (+001.44%)	+7-3	32 17
Cnr6	0.8649 – 0.8831 (+002.10%) 0.0130 – 0.0568 (+006.22%)	1.0000 – 0.9844 (-001.56%) 0.0000 – 0.0085 (+000.85%)	+9-1	26 7
Cnr7	0.8649 – 0.9140 (+005.65%) 0.0130 – 0.0344 (+002.91%)	1.0000 – 0.9749 (-002.50%) 0.0000 – 0.0204 (+002.04%)	+10	28 7
Cnr8	0.8649 – 0.8739 (+001.08%) 0.0130 – 0.0280 (+003.38%)	1.0000 – 0.9857 (-001.43%) 0.0000 – 0.0198 (+001.98%)	+10	23 5

Tabela A.9: Grupo 2 – Função de avaliação multi-objetivo – *Fnc2*

Apêndice

B

Resultados Experimentais — Gráficos

B.1 Grupo 1

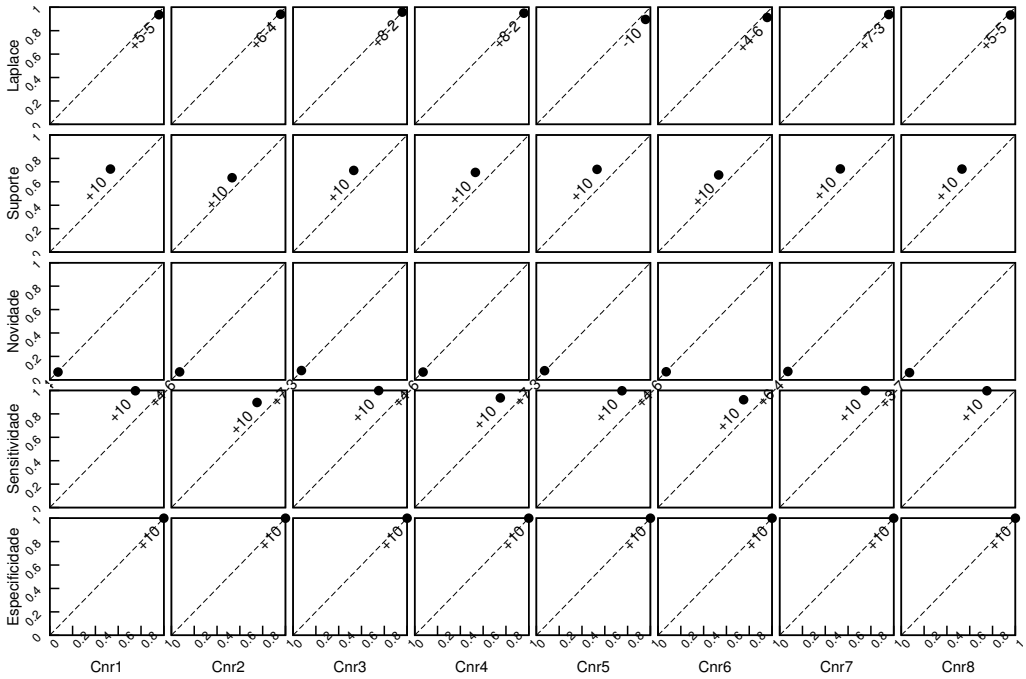

Figura B.1: Grupo 1 – breast – Relação entre os valores médios inicial e final das funções de avaliação simples-objetivo

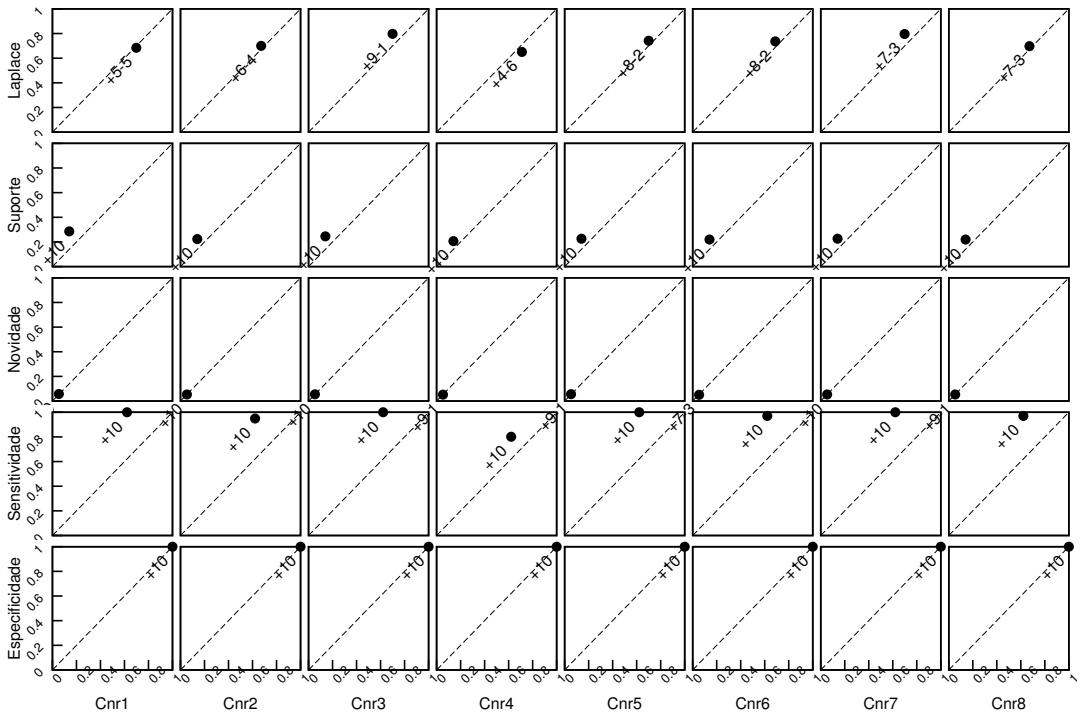

Figura B.2: Grupo 1 – cmc – Relação entre os valores médios inicial e final das funções de avaliação simples-objetivo

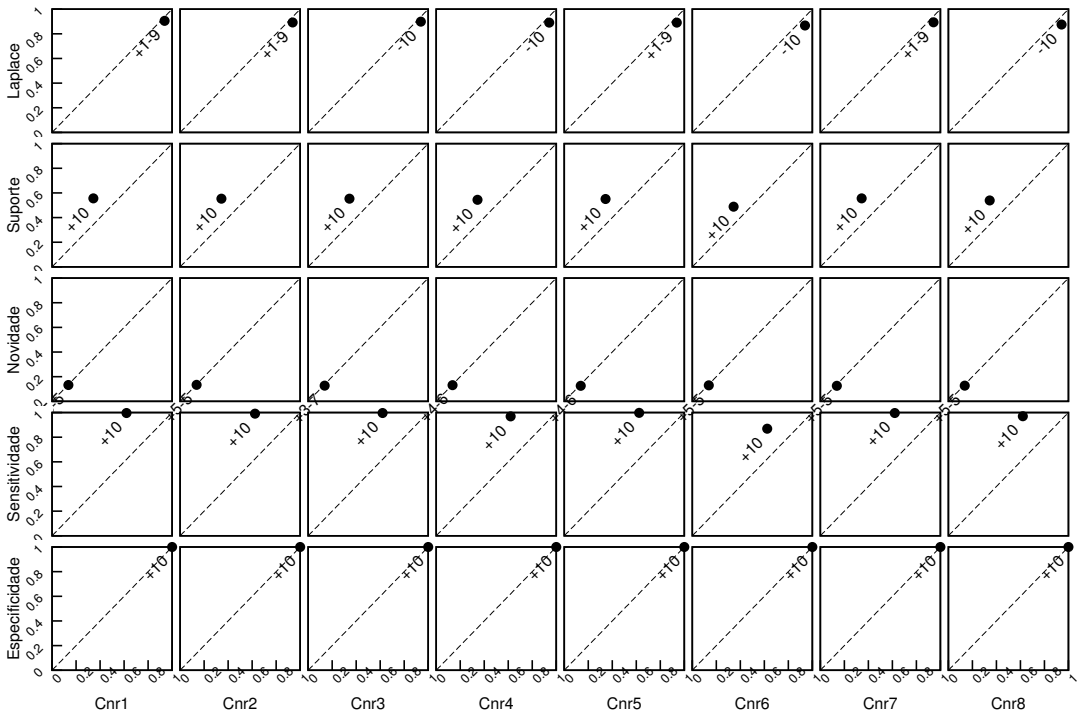

Figura B.3: Grupo 1 – heart – Relação entre os valores médios inicial e final das funções de avaliação simples-objetivo

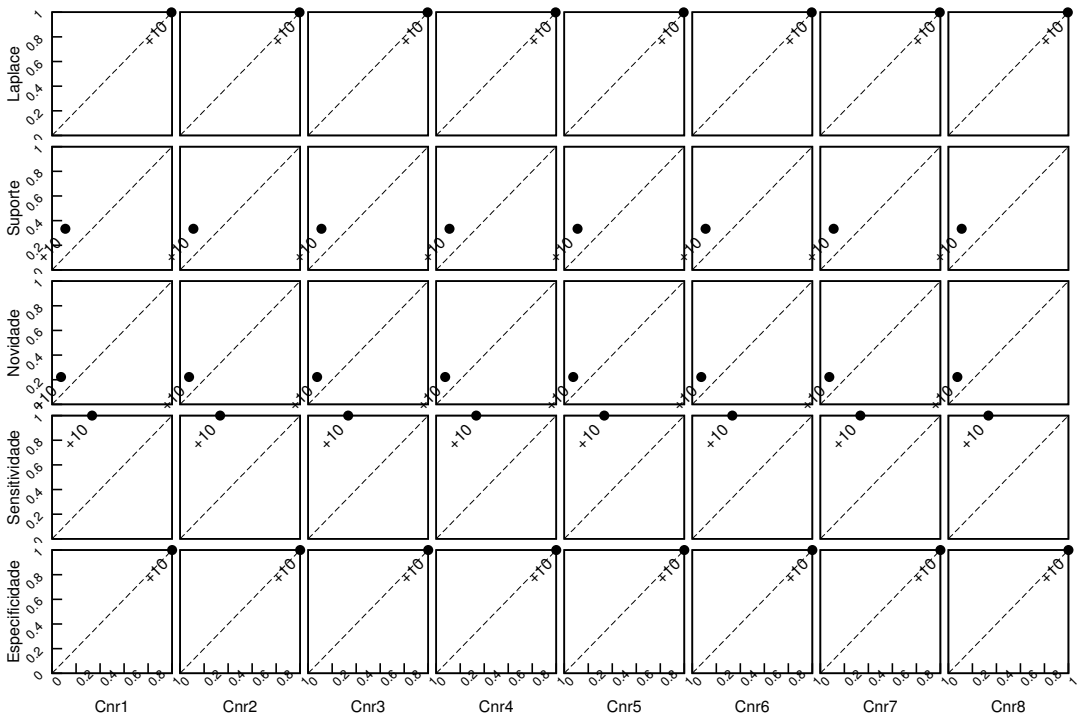

Figura B.4: Grupo 1 – nursery – Relação entre os valores médios inicial e final das funções de avaliação simples-objetivo

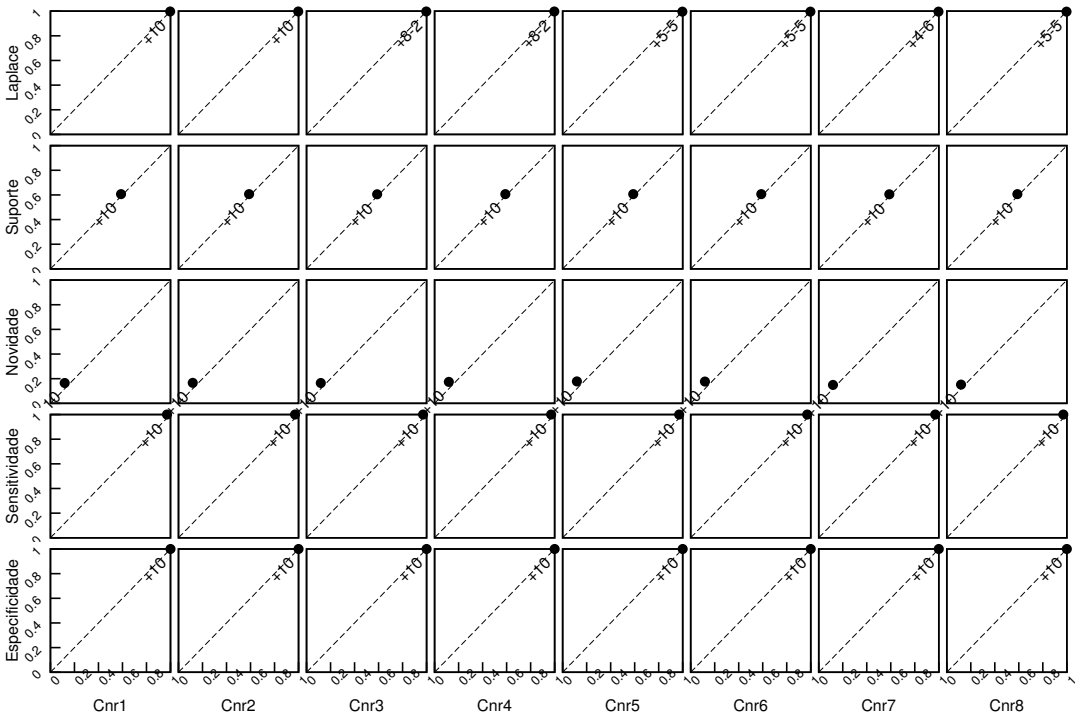

Figura B.5: Grupo 1 – spambase – Relação entre os valores médios inicial e final das funções de avaliação simples-objetivo

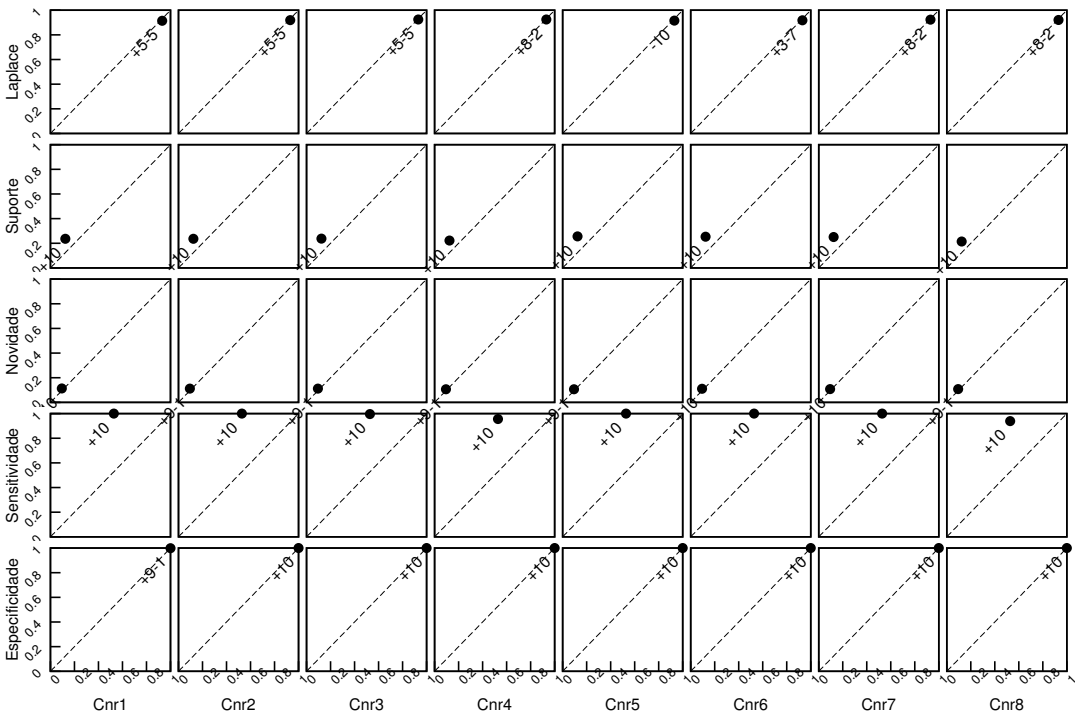

Figura B.6: Grupo 1 – vehicle – Relação entre os valores médios inicial e final das funções de avaliação simples-objetivo

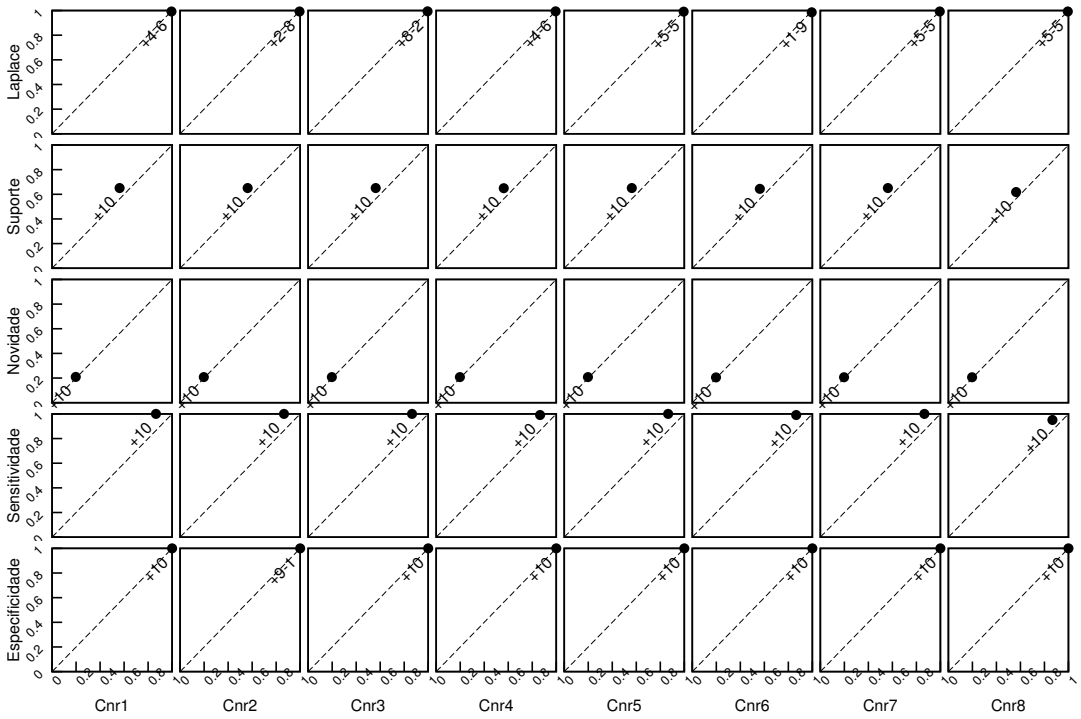

Figura B.7: Grupo 1 – wbc – Relação entre os valores médios inicial e final das funções de avaliação simples-objetivo

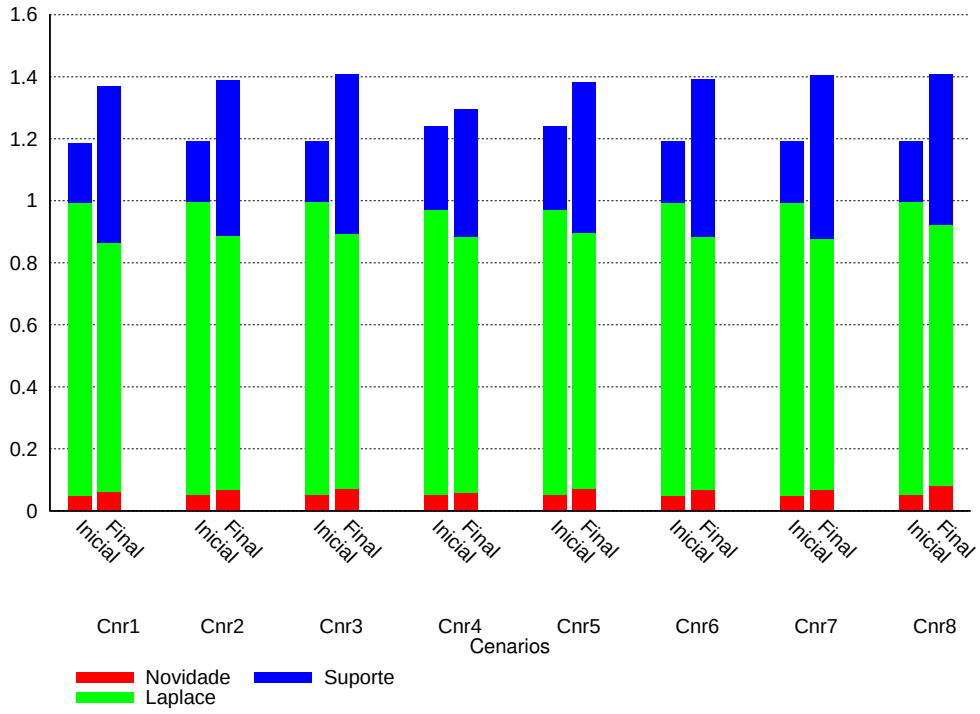

Figura B.8: Grupo 1 – breast – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

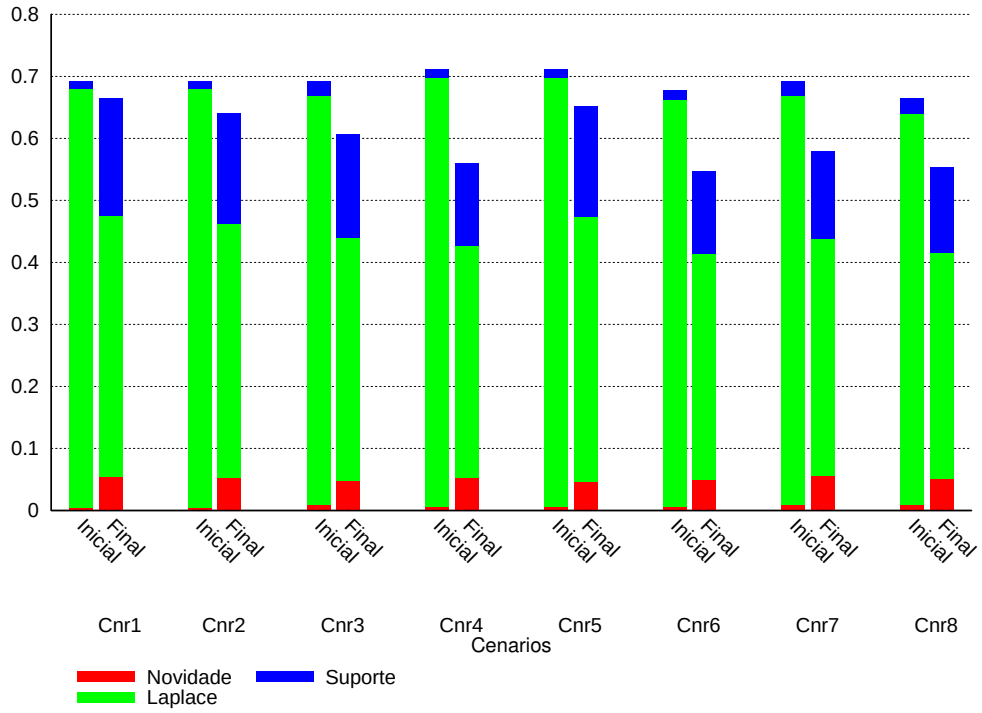

Figura B.9: Grupo 1 – cmc – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

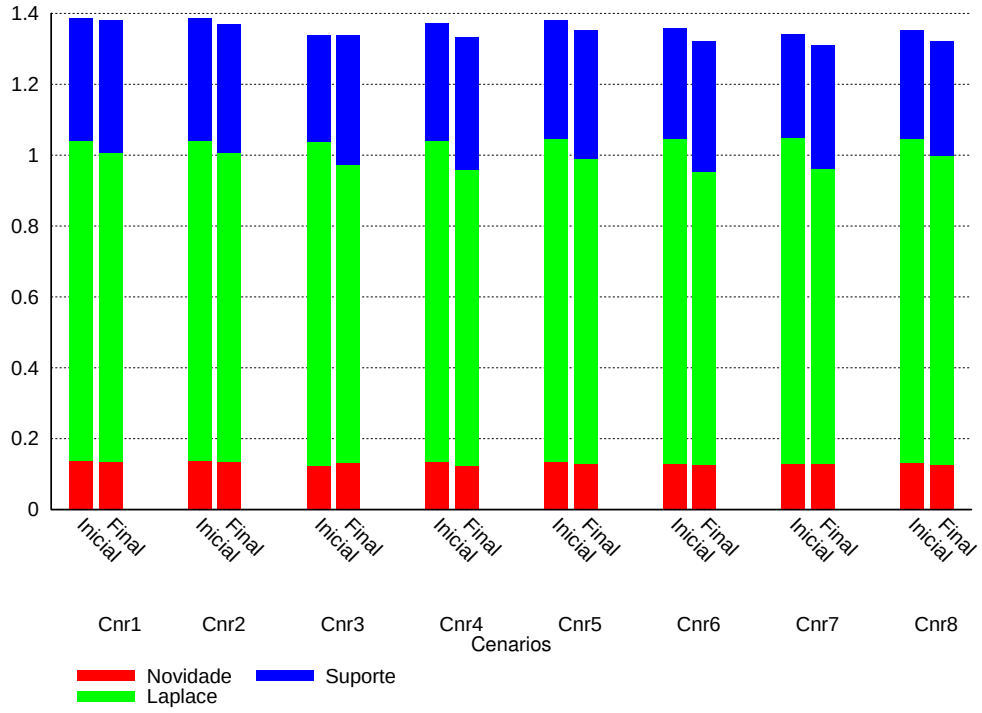

Figura B.10: Grupo 1 – heart – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

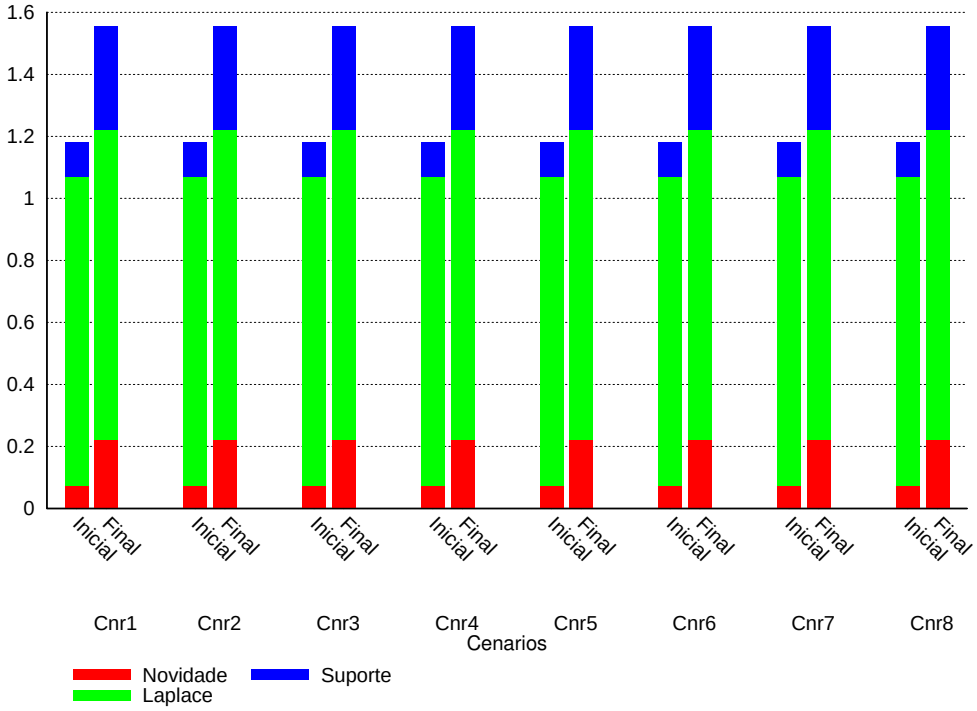

Figura B.11: Grupo 1 – nursery – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

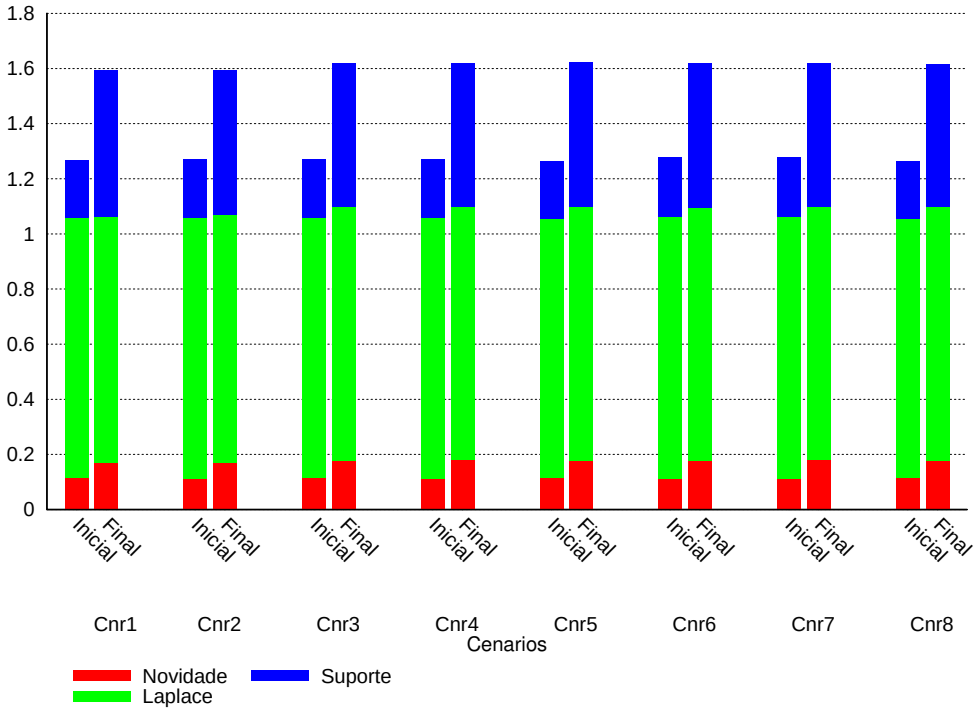

Figura B.12: Grupo 1 – spambase – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

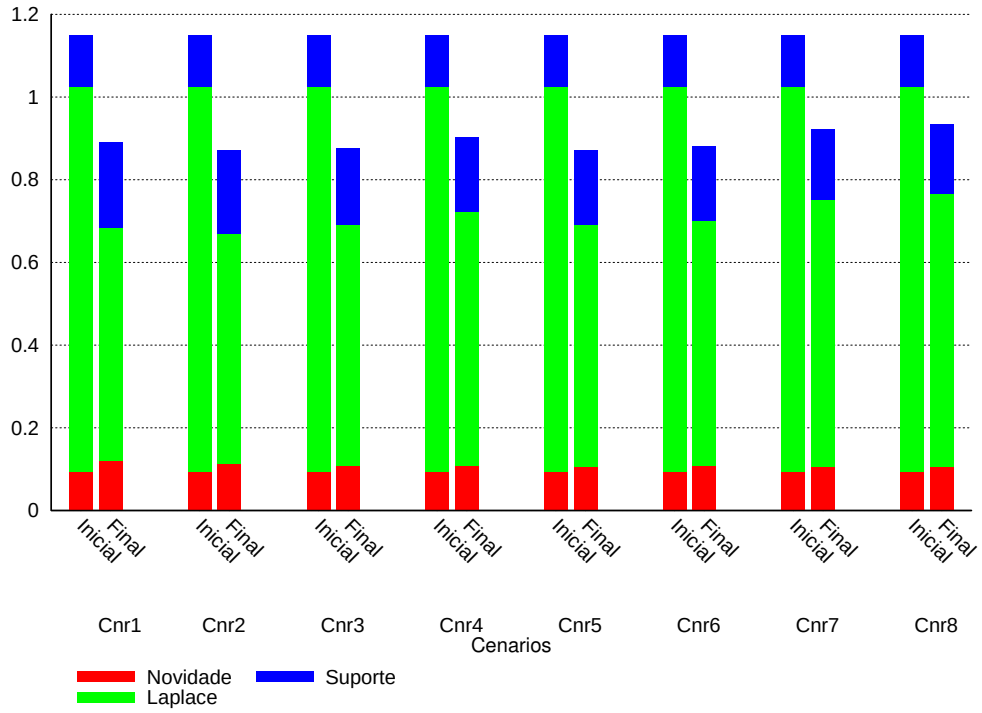

Figura B.13: Grupo 1 – vehicle – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

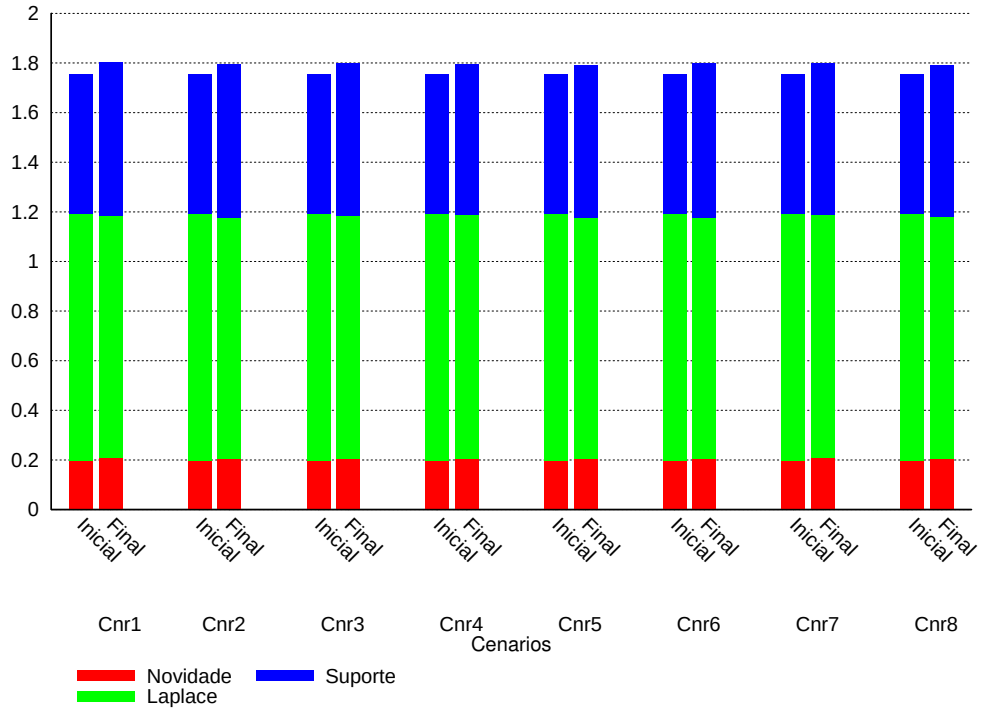

Figura B.14: Grupo 1 – wbc – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

B.2 Grupo 2

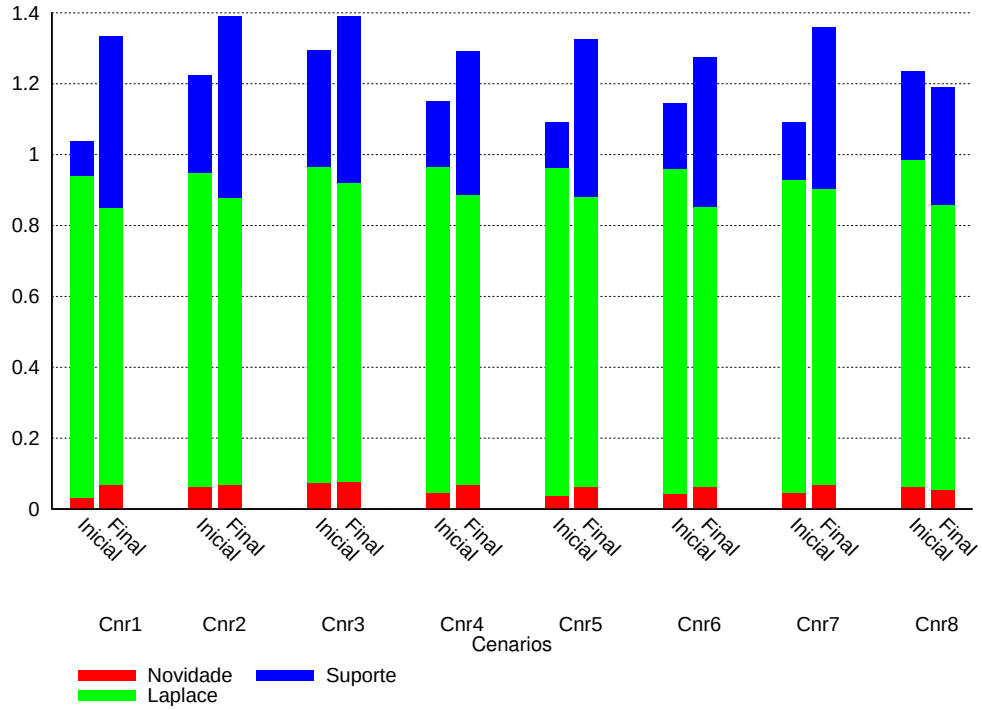

Figura B.15: Grupo 2 – breast – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

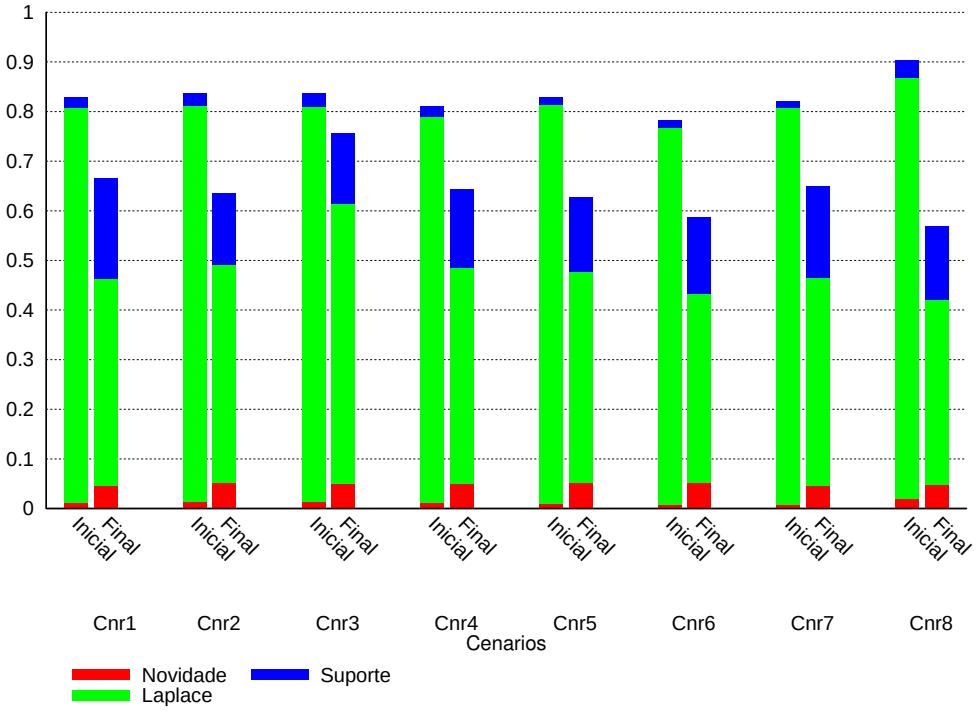

Figura B.16: Grupo 2 – cmc – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

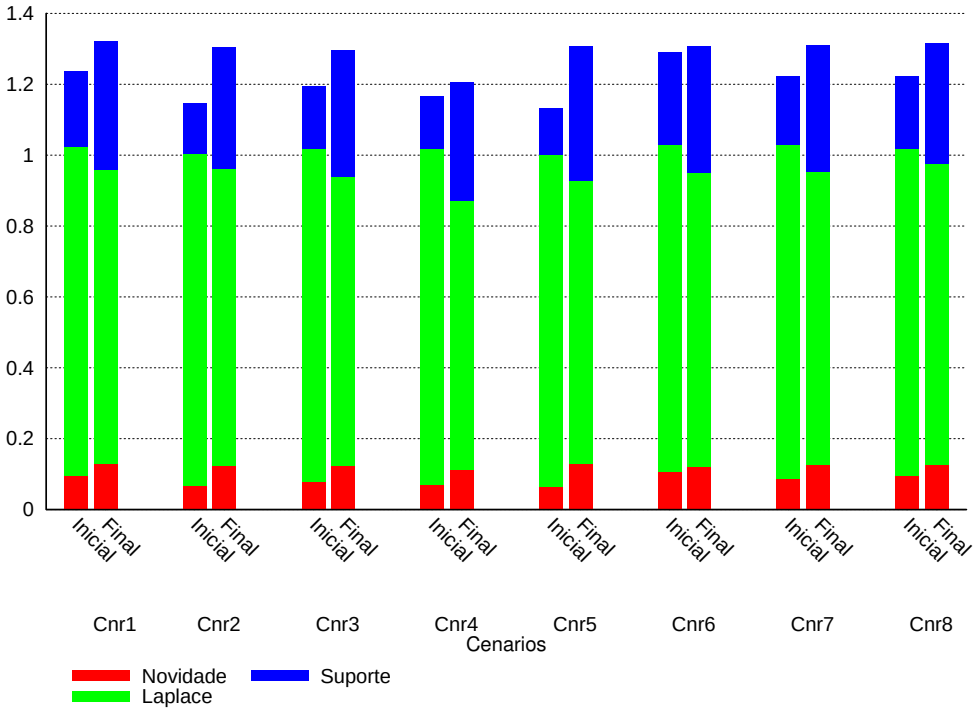

Figura B.17: Grupo 2 – heart – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

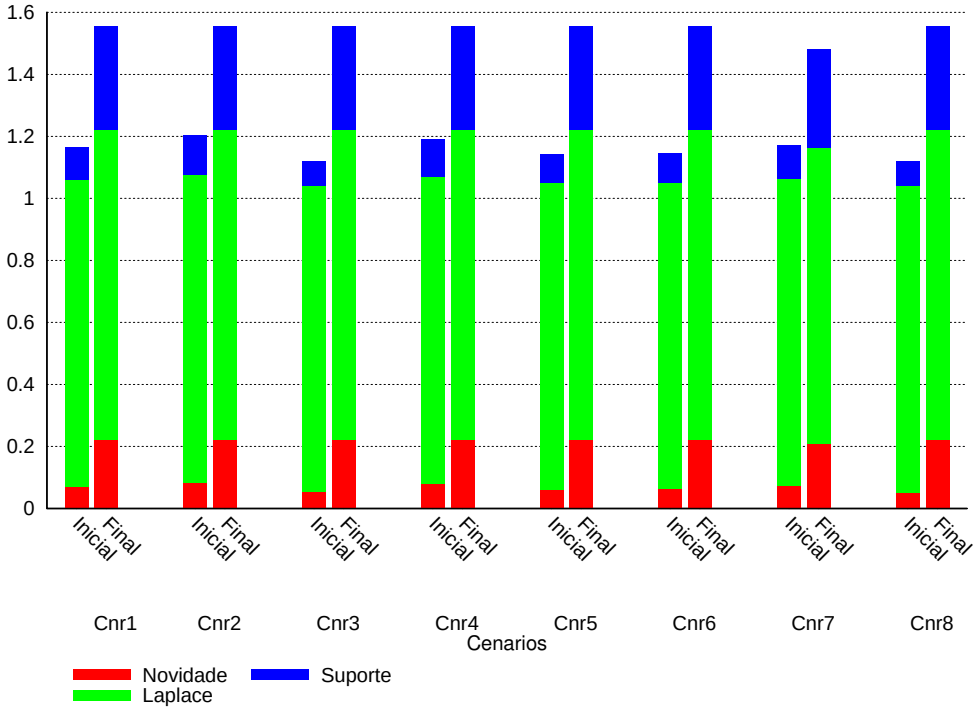

Figura B.18: Grupo 2 – nursery – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

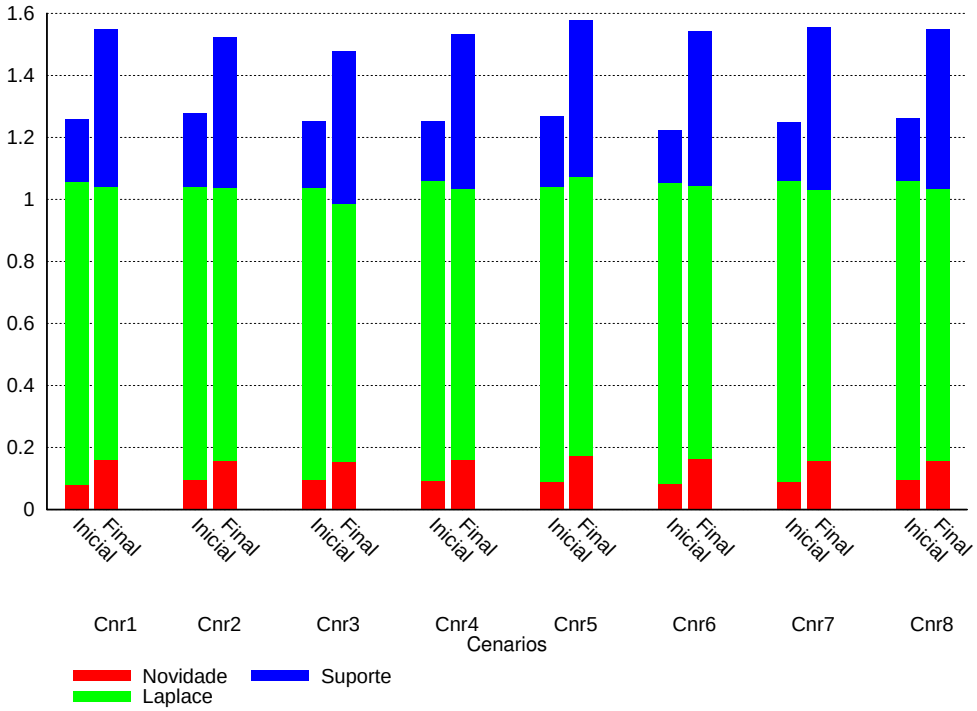

Figura B.19: Grupo 2 – spambase – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

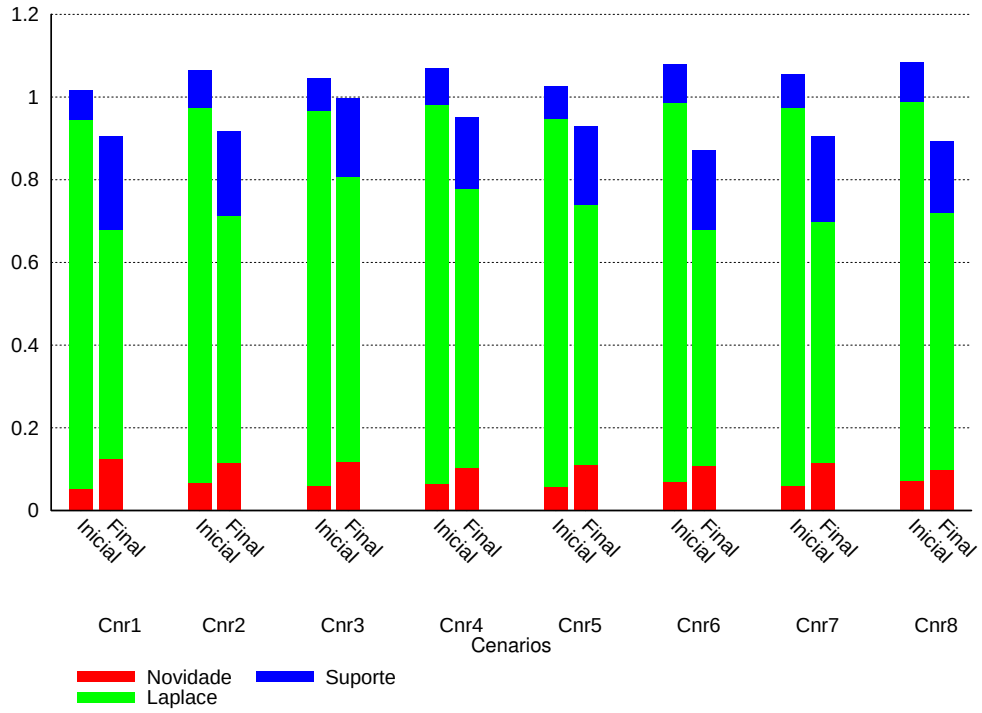

Figura B.20: Grupo 2 – vehicle – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*

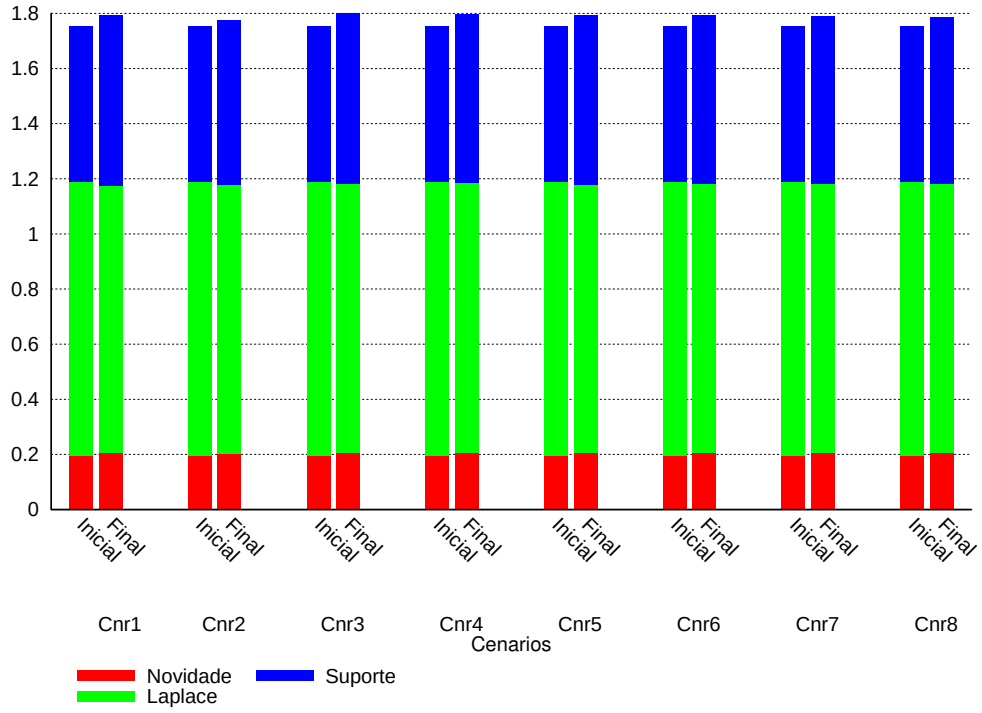

Figura B.21: Grupo 2 – wbc – Valores médios inicial e final das medidas que compõem a função multi-objetivo *Fnc1*