Rafael Abud Menezes

CONSTRUÇÃO DE CLASSIFICADORES HIERÁRQUICOS MULTIRRÓTULO USANDO EVOLUÇÃO DIFERENCIAL

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

CURITIBA

2014

Rafael Abud Menezes

CONSTRUÇÃO DE CLASSIFICADORES HIERÁRQUICOS MULTIRRÓTULO USANDO EVOLUÇÃO DIFERENCIAL

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

Área de Concentração: *Descoberta do conhecimento e aprendizagem de máquina*

Orientador: Prof. Dr. Júlio Cesar Nievola

CURITIBA

2014

Menezes, Rafael Abud
Construção de classificadores hierárquicos multirrótulo usando Evolução Diferencial. Curitiba, 2014. 139p.

Tese (Doutorado) – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. Classificadores hierárquicos 2. Multirrótulo 3. DAG 4. Proteína. I. Nievola, Júlio Cesar. II. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática.

Sumário

Lista de figuras

Lista de tabelas

Resumo

O tipo de classificador usado para se definir as classes de um determinado objeto depende da forma como as possíveis classes do objeto se relacionam e do número de classes as quais o objeto pode pertencer. Os classificadores hierárquicos tem sido usados para a classificação de textos e, mais recentemente, para a classificação da função de proteínas, tarefa dificultada pelo fato de uma proteína poder pertencer a mais de uma função e de as possíveis relações terem relação de hierarquia entre si. As bases de dados usadas para a construção de classificadores de proteínas podem ser compostas por instâncias que executam uma função, com o problema sendo chamado de monorrótulo ou de rótulo simples, e por instâncias que executam uma ou mais funções, com o problema sendo chamado de multirrótulo. Além disso, a hierarquia entre as classes pode ser representada por uma árvore ou por um grafo (DAG). Devido às dificuldades citadas e ao fato de as proteínas serem fundamentais para o funcionamento do organismo de qualquer ser vivo, as pesquisas acerca das suas funções se apresentam como uma das mais atuantes na bioinformática. Como as funções das proteínas podem ser representadas computacionalmente por árvores ou DAG´s, classificadores locais e globais podem ser construídos. O objetivo principal deste trabalho é a construção de um método, chamado RCM e baseado na Evolução Diferencial, para a construção de classificadores hierárquicos multirrótulo para a definição da função de proteínas para uma hierarquia de classes representada por um DAG. Assim, a Evolução Diferencial, que foi proposta inicialmente para a resolução de problemas de otimização com atributos contínuos, é usada aqui em problemas de otimização com atributos discretos. Como parte do desenvolvimento do RCM, outros sete métodos são propostos, variando 1) a estrutura de dados usada para representar a hierarquia entre as classes, 2) a forma como a estrutura de classes é explorada para se realizar a classificação e 3) o número de classes as quais um exemplo pode pertencer. Uma das contribuições deste trabalho é a possibilidade de utilização do método independente das características do problema apresentado. Nos experimentos realizados, todas as versões do RCM se mostraram competitivas quando comparadas a outros métodos disponíveis na literatura. Dessa forma, o RCM se mostrou como uma alternativa para a construção de classificadores hierárquicos de proteínas. Dessa forma, pode-se concluir que a Evoluçñao Diferencial mostrou ter potencial para ser usada em problemas de otimização com atributos discretos.

Abstract

The classifier used to label an object depends on the relationships among the classes that the objects can belong to and on the number of classes that one object can belong to. The hierarchical classifiers have been used with the objective of classifying texts and, recently, with the objective of classifying proteins. The task of classifying proteins has mainly two difficulties: the first is that one protein can belongs to one or more functions. The second is that the classes that the proteins can belong to have hierarchical relationshop among them. The datasets used to the building of hierarchical classifiers of proteins can be composed by instances that perform one function, where the problem is called single-label and can be composed by instances that perform one or more functions, where the problem is called multi-label. Moreover, the hierarchy among the classes can be represented by a tree or a DAG. Due to the difficulties cited and the fact that the proteins perform the main tasks in the organism of any live being, the research about the function of proteins is one of the most active areas in bioinformatics. Since the function of proteins can be represented computationally by tree or DAG´s, local and global classifiers can be built. In this work a method based on Differential Evolution and called RCM is proposed. The main objective of RCM is the building of global hierarchical classifiers with the relationship among the classes represented by a DAG. Thus, the Differenrital Evolution, which is proposed to solve optimization problems with continuous atributes, here is used to solve a problem with discrete atributes. In the process of development of RCM, other seven methods are proposed, varying 1) the structure of classes used to represent the relationships among the classes, 2) the way that the structure of classes is explored during the procces of and 3) the number of classes that an example can belongs to. One of the contribution of this work is the possibility of utilization of the method independently of the features of the problem. In the experiments, all of the versions of RCM presented competitive results when compared with other methods avaiable in the literature with the same objective. In this way, based on the results presented in this work, the Differential Evolution can be used in the solution of problems with discrete atributes.

Lista de abreviaturas

DAG – Directed Acyclic Graph

RCM – Rule Construction Method

ED – Evolução Diferencial

GO – Gene Ontology

FunCat – Functional Catalogue

PSO – Particle Swarm Optimization

ACO- Ant Colony Optimization

Capítulo 1

Introdução

Os algoritmos utilizados para mineração de dados extraem conhecimento a partir de bases de dados. Com o conhecimento extraído, principalmente, duas tarefas podem ser realizadas com esses dados. A primeira delas é o agrupamento, no qual o algoritmo agrupa os dados da base, colocando cada um deles em um grupo diferente. Nesse caso o algoritmo deve criar os grupos, ou *clusters*, e definir a qual deles cada um dos exemplos pertence. A segunda é a classificação, na qual cada um dos exemplos da base pertence a uma classe e o algoritmo cria um modelo para a definição da classe à qual os novos exemplos pertencem. Esse modelo é criado com base em exemplos do objeto já classificados para cumprir o seu objetivo. A construção de classificadores hierárquicos de proteínas é, atualmente, uma das tarefas mais estudadas na bioinformática [Salama and Freitas, 2013].

1.1 Descrição do problema

As proteínas executam funções dentre as mais importantes no corpo de um ser vivo. Elas podem exercer o papel de um receptor transmembranar, por meio das proteínas GPCR, de um catalisador de reações químicas e biológicas, por meio das enzimas, de um transportador extracelular, por meio da hemoglobina e mioglobulina, e de protetor, por meio da imunoglobulina, hormonal, por meio da insulina, dentre outros [Devlin, 2011].

Conhecer a função das proteínas pode ajudar os profissionais das ciências da saúde, por exemplo, a encontrar vias para melhorar a qualidade de vida do ser humano. O conhecimento acerca da insulina pode ser usado para o tratamento e consequente aumento da qualidade de vida dos diabéticos, por exemplo.

As funções das proteínas têm relação de dependência entre si, podendo assim ser organizadas hierarquicamente. Computacionalmente, essas funções podem ser representadas por uma árvore, como mostrado na figura 1.1, ou por um grafo (DAG), como mostrado na figura 1.2. Na árvore da figura 1.1, o nó associado à classe 1 é pai

dos nós associados às classes 1.1 e 1.2, enquanto o nó associado à classe 2 é pai dos nós associados às classes 2.1 e 2.2. Conforme mostrado na figura 1.2, computacionalmente, a classe 1.1.1-2.1.1.1 e suas dependências podem ser representadas por um grafo.

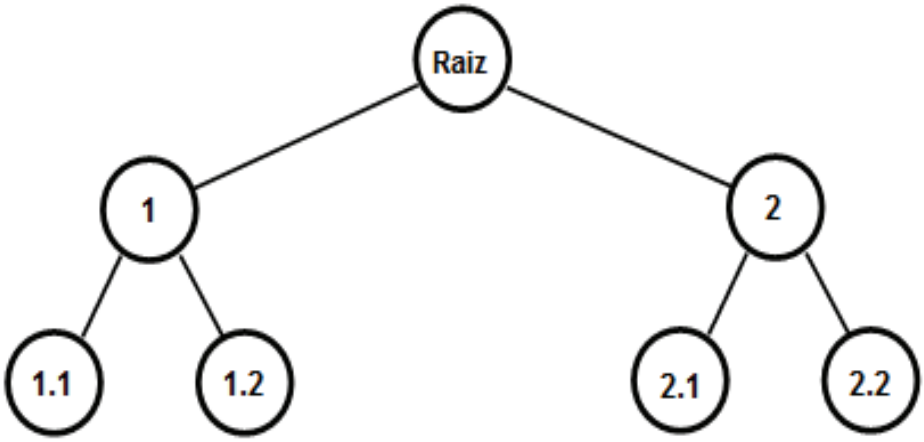

Figura 1.1 Representação de funções proteicas organizadas em uma árvore.

Conforme ilustrado nas figuras 1.1 e 1.2, cada nó em uma árvore pode ter apenas um nó-pai e cada nó em um grafo pode ter mais de um pai.

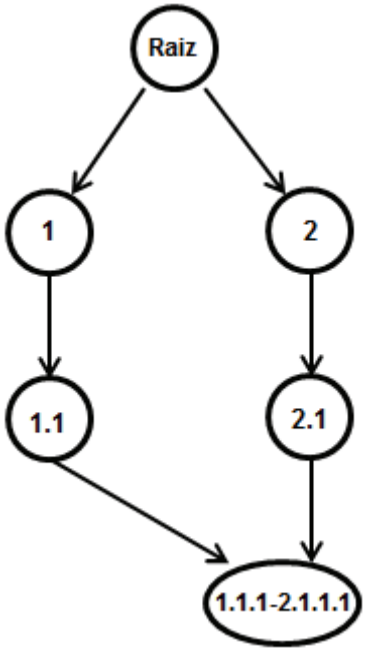

Figura 1.2 Representação da função 1.1.1-2.1.1.1 e suas dependências.

Milhares de proteínas têm sido descobertas nos mais diversos projetos sobre o sequenciamento do genoma. Porém, a função dessas proteínas ainda é desconhecida, o que gerou uma necessidade de se classificar proteínas de forma rápida e eficiente.

As possíveis funções das proteínas estão organizadas em uma estrutura hierárquica com milhares de nós, sendo que uma proteína pode exercer mais de uma função e por isso estar associada a mais de um nó dessa estrutura.

Para superar as dificuldades citadas, a biologia pode ser auxiliada por técnicas computacionais que auxiliam outras áreas do conhecimento como a Química [Kreutz et al., 2010] e a Medicina [Maulik, 2009], por exemplo. O método proposto neste trabalho, chamado RCM (do inglês Rule Construction Method), é um gerador de classificadores globais multirrótulo de funções proteicas para uma hierarquia de classes representada por um DAG. Dessa forma, um algoritmo baseado na Evolução Diferencial (ED) é usado para a geração de classificadores hierárquicos compostos por regras do tipo SE condições ENTÃO conclusão.

As regras que compõem o classificador são geradas a partir de uma base de dados já rotulada, sendo que essas regras se baseiam nos atributos dos exemplos presentes na base. Por isso, um conjunto de proteínas para as quais já se conhece a classificação deve estar disponível e armazenado em uma base de dados. A descrição das classes e das suas relações devem estar descritas em uma ontologia. As informações acerca das relações entre as funções são usadas para o cálculo das medidas de avaliação do classificador.

O classificador gerado pelo método proposto poderá ser usado por profissionais das ciências biológicas para o auxílio na classificação da função de novas proteínas. Dessa forma, ao se descobrir uma nova proteína, assim que as características da nova proteína solicitada pelo classificador forem conhecidas, o classificador usará essas características para predizer a (s) função (ões) da proteína descoberta.

1.2 Motivação

A Evolução Diferencial foi criada com o objetivo de ser usada para a resolução de problemas de otimização com valores contínuos[Storn, 1996]. Quando usada em comparação com outras técnicas de mesmo objetivo na resolução desse tipo de problema, a ED se mostra como a técnica de melhor desempenho [Veterstrom and Thomsen, 2004]. A principal motivação deste trabalho é o uso da ED para a criação de classificadores das funções de proteínas, que é um problema com valores discretos. Como as proteínas executam as mais importantes funções no organismo de qualquer ser

vivo, a construção de uma ferramenta que defina a função de novas proteínas descobertas se torna também uma motivação, já que aumentaria a qualidade de vida, principalmente, dos seres humanos.

1.3 Hipótese

A ED, assim como tem sido usada para problemas de otimização com valores contínuos, pode ser usada para a geração de classificadores hierárquicos de proteínas com valores discretos. Os classificadores gerados podem ser locais ou globais, monorrótulo ou multirrótulo e usando árvores ou DAG´s para representar a hierarquia entre as classes.

1.4 Contribuições

As contribuições deste trabalho são:

-O uso da ED em problemas com valores discretos.

-O uso da ED para a geração de classificadores hierárquicos de proteínas, independente da forma como o problema se apresenta, podendo variar na estrutura computacional usada para se representar a hierarquia entre as classes (Árvore ou DAG), na maneira como a estrutura é explorada para se definir a classe a qual a nova instância pertence (local ou global) e no número de funções que cada uma das instâncias pode pertencer (monorrótulo ou multirrótulo).

-A criação de uma estratégia para mutação de classes hierárquicas para a ED quando essas classes são representadas por sub-árvores na geração de classificadores globais.

1.5 Objetivos

O objetivo principal deste trabalho é a construção e avaliação de um método gerador de classificadores hierárquicos globais multirrótulo das funções de proteínas para uma hierarquia entre as classes representada por um DAG. Para se chegar ao objetivo principal, dois objetivos específicos devem ser cumpridos, na seguinte ordem:

1-Avaliar do uso da ED em problema com valores discretos e para a construção de classificadores hierárquicos.

2- Construir e avaliar um método gerador de classificadores hierárquicos globais multirrótulo para uma hierarquia de classes representada por um DAG usando a ED.

1.7 Organização

A metodologia está descrita no capítulo 2, enquanto a fundamentação teórica está descrita no capítulo 3. No capítulo 4 estão descritos os principais trabalhos usando a classificação hierárquica de proteínas. No capítulo 5 está descrito o RCM e o seu processo de desenvolvimento. No capítulo 6 está a descrição dos experimentos feitos. No capítulo 7 estão descritos os resultados e no capítulo 8 estão as conclusões deste trabalho.

1.8 Considerações finais deste capítulo

Neste capítulo foram descritos os componentes necessários para o entendimento do trabalho. Inicialmente, foi descrito o problema a ser tratado neste trabalho e as suas principais dificuldades. Posteriormente, foi descrita as principais motivações para a realização do trabalho, a hipótese, as principais contribuições e os objetivos deste trabalho. Por fim, é descrito como o documento está organizado.

No próximo capítulo está descrita a metodologia usada para o desenvolvimento deste trabalho. O conhecimento desse processo é importante para que o uso de cada um dos componentes descritos nos capítulos restantes seja entendido.

Capítulo 2

Metodologia

Etapa 1

Inicialmente, foi feita uma pesquisa na literatura sobre o estado da arte da ED para que a versão da ED com o menor número de limitações possível fosse usada no RCM. Foram estudadas as versões da ED com os objetivos mais semelhantes aos objetivos do RCM. Para isso foi criada uma versão preliminar do RCM com o uso de uma das versões originais da ED. A partir dos problemas apresentados verificou-se aqueles que foram gerados devido ao uso da estratégia da ED escolhida. Assim, procurou-se uma versão disponível na literatura que solucionasse os problemas apresentados.

Etapa 2

Posteriormente, foram estudados os métodos com os mesmos objetivos de cada uma das oito versões do RCM. Para as escolhas dos métodos a serem estudados foi levada em consideração, além do objetivo e características dos métodos, a data de publicação.

Etapa 3

O teste do uso da ED para a geração de classificadores hierárquicos das funções proteicas foi feito por meio do método para construção de classificadores locais de rótulo simples para uma hierarquia entre as classes representada por uma árvore. O método foi desenvolvido levando-se em consideração as versões da ED estudadas e comparado com outro de mesmo objetivo, dentre aqueles estudados na etapa anterior. Assim, o método foi construído por meio do uso de uma base de treino e avaliado por meio de uma base de teste.

Etapa 4

Após se atestar que o uso da ED para a construção de classificadores hierárquicos das funções de proteínas é viável (por meio da primeira versão do RCM), novas versões do método foram desenvolvidas, sendo a seguinte mais complexa do que

a anterior. Isso foi feito para que com uma versão mais simples as limitações do método pudessem ser minimizadas antes do desenvolvimento de uma versão mais complexa. Assim, com a primeira versão desenvolvida e os seus problemas identificados e solucionados, pôde-se construir a sua versão global, sendo assim a segunda versão do RCM. Com os problemas da segunda versão identificados, pôde-se desenvolver duas versões (terceira e quarta) do RCM com a hierarquia de classes sendo representadas de maneira mais complexa do que nas versões anteriores (por um DAG). Após isso, os quatro construtores de classificadores hierárquicos multirrótulo foram desenvolvidos de maneira semelhante às quatro primeiras versões do RCM. Assim, inicialmente, foi desenvolvido um método construtor de classificadores hierárquicos locais para uma hierarquia de classes representada por uma árvore. O próximo passo foi o desenvolvimento de um método para a construção de classificadores globais multirrótulo para uma hierarquia de classes representada por uma árvore, seguido pelo desenvolvimento das versões que constroem classificadores multirrótulo globais e locais de maneira mais complexa, com a hierarquia entre as classes sendo representada por um DAG.

Cada método foi desenvolvido por meio de uma base de treino, avaliado por meio de uma base de teste e comparado com um método estudado na etapa 2 de mesmo objetivo.

Capítulo 3

Fundamentação Teórica

Neste capítulo são descritos os principais conceitos relacionados à construção e entendimento do RCM. Na seção 3.1 são apresentados os classificadores, que são o resultado da execução do método aqui proposto. Na seção 3.3 descreve-se a Evolução Diferencial, algoritmo usado como base para execução do RCM. Na seção 3.4 tem-se uma introdução às proteínas, objeto a ter suas funções definidas pelos classificadores gerados pelo RCM.

3.1 Classificadores

Um algoritmo usado para executar uma tarefa de classificação passa, principalmente, por duas fases. No caso de um algoritmo usado para a criação de regras de classificação, a fase de treinamento é aquela em que as regras são criadas, formando um conjunto de regras e, consequentemente, um classificador. Nessa fase, os exemplos usados para a construção das regras compõem a base de treinamento ou o conjunto de treinamento. Após a fase de treinamento, o classificador é avaliado a partir de outra base de dados, chamada de base de testes ou conjunto de testes.

Os classificadores buscam suprir a necessidade de se classificar instâncias de forma mais rápida e menos custosa, usando apenas as características dos objetos como critério para classificação. A classificação pode ser definida como a tarefa de mapear uma variável (classe) a partir dos valores de outras variáveis (atributos) [Costa et al., 2007], e pode ser pode ser discriminativa ou probabilística. No modelo discriminativo, um padrão de classificação é usado para, a partir dos atributos, encontrar a classe do objeto. Já no modelo probabilístico é calculada uma probabilidade de o objeto pertencer a cada uma das classes [Hand et al., 2001].

Os classificadores bayesianos são um exemplo de classificadores probabilísticos [Silla and Freitas, 2009]. Nesse modelo, caso os antecedentes sejam verdadeiros, os consequentes tem uma determinada probabilidade de acontecer. O tipo de classificador criado depende da forma como as classes das instâncias se relacionam entre si. Portando, os classificadores podem ser hierárquicos ou não hierárquicos. Os classificadores hierárquicos, gerados pelo RCM, são descritos na seção 3.1.1.

3.1.1 Classificadores hierárquicos

Os exemplos presentes em uma base de dados podem pertencer a classes independentes entre si ou podem pertencer a classes com relação de hierarquia ou dependência umas com as outras. A existência de relação entre as classes definem o tipo de classificador usado para se rotular novos exemplos do objeto em questão. Para os casos onde não há dependência entre as classes, o classificador usado é chamado não hierárquico. Para os outros casos o classificador usado é chamado hierárquico [Silla and Freitas, 2009].

Algumas proteínas executam determinadas funções como "Transporte celular" ou "Síntese de substâncias" em um órgão. Alguns exemplos dessas funções, que possuem relação de hierarquia entre si, são listadas na ultima colunada tabela 3.1. A proteína que tem a função de realizar o "Transporte Intracelular", por exemplo, têm a função de realizar o "Transporte" (em geral, em qualquer órgão).

Tabela 3.3.1 Exemplo de uma base de dados com exemplos de proteínas classificadas tendo as classes relação de dependência entre si.

Exemplo/Atributo	Nº lig. Peptídicas	Peso molecular	Nº de mol. de Nitr.	Função
1	6	88.5	8	Transporte
2	4	45.0	15	Síntese renal
3	2	54.2	13	Transporte extra-celular
4	9	12.1	5	Transporte
5	7	99.9	2	Síntese
6	6	86.5	19	Transporte intra-celular
7	4	70.9	11	Síntese hepática

Assim, proteínas que pertencem à classe “Transporte intracelular” também pertencem à classe “Transporte”. Isso torna a classe “Transporte intracelular” relacionada à classe “Transporte”. A classe “Transporte extracelular” também é relacionada à classe “Transporte”, assim como as classes “Síntese hepática” e “Síntese renal” são relacionadas à classe “Síntese”.

As classes com relação de dependência ou hierarquia entre si podem ser representadas computacionalmente por uma árvore ou por um DAG [Costa et al., 2007] [Metz, et al, 2011], sendo que, independentemente da estrutura, cada nó representa uma classe e as arestas representam as relações entre as classes relacionadas. A hierarquia existente entre as classes da tabela 3.1 pode ser representada por uma árvore, conforme ilustrado na figura 3.1.

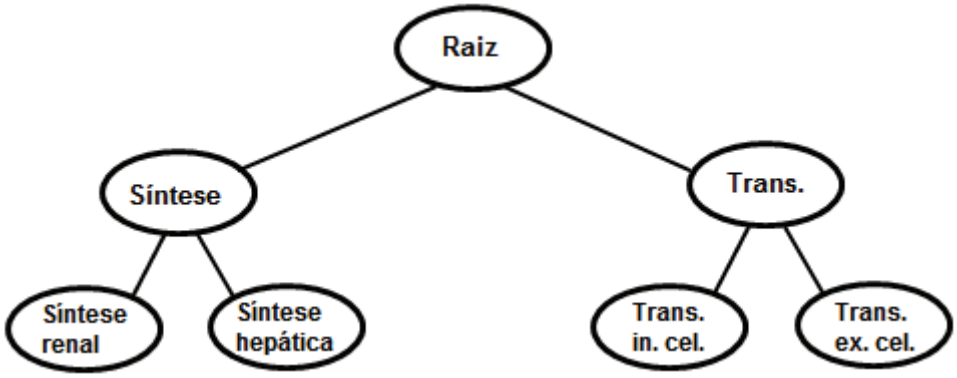

Figura 3.1 Árvore que representa a relação de dependência entre as classes da tabela 3.1.

Na árvore da figura 3.1, os nós que representam as classes “Síntese renal” e “Síntese hepática” são nós-filhos do nó que representa a classe “Síntese”. Assim, as classes “Síntese renal” e “Síntese hepática” são classes filhas (e por isso, descendentes) da classe “Síntese”. Caso as classes “Síntese renal” e “Síntese hepática” tivessem filhas, elas também seriam descendentes da classe “Síntese”. Os nós que representam as classes “Transporte intracelular” e “Transporte extracelular” são nós-filhos do nó que representa a classe “Transporte” e as classes “Transporte intracelular” e “Transporte extracelular” são classes descendentes da classe “Transporte”. Os nós que representam as classes “Síntese renal”, “Síntese hepática”, “Transporte intracelular” e “Transporte extracelular”, por não terem nós-filho, são chamados nós-folha.

Para classificar objetos em que as possíveis classes têm relação hierárquica entre si a classificação pode ser plana ou hierárquica. Quando as relações entre as classes são

ignoradas e apenas os nós-folha são usados, se diz que é feita classificação plana. Quando toda a estrutura de classes é levada em consideração, se diz que é feita classificação hierárquica [Freitas and de Carvalho, 2007] [Silla and Freitas, 2011] [Costa et al., 2007]. Para a explicação dos classificadores hierárquicos a seguir, será usada a base de dados descrita na tabela 3.2, onde são listados sete exemplos, cada um com três atributos fictícios (Atributo A, Atributo B, Atributo C). Dentre as classes dos exemplos descrito na tabela 3.2, os nós 1.1 e 1.2 são nós-filhos do nó 1, enquanto os nós 2.1 e 2.2 são nós-filhos do nó 2. Os nós 1.1, 1.2, 2.1 e 2.2 são nós-folha.

Tabela 3.2 Exemplo de uma base de dados com exemplos de proteínas classificadas com as classes relacionadas entre si.

Exemplo/Atributo	Atributo A	Atributo B	Atributo C	Classe
1	9	5	8	1.2
2	4	3	6	1.1
3	3	1	2	1
4	9	7	2	2.1
5	5	6	3	2.2
6	7	5	4	2
7	9	8	1	1.2

A hierarquia das classes da base descrita na tabela 3.2 pode ser representada pela árvore ilustrada por meio da figura 3.2. Como pode ser observado, os exemplos 1, 2, 3 e 7 pertencem à mesma classe no nível 1 da estrutura usada para representar a hierarquia entre as classes e, por isso, as classes às quais os exemplos pertencem estão relacionadas entre si. Da mesma forma, os exemplos 4,5 e 6 pertencem à mesma classe no nível 1 da estrutura de classes usada para representar a hierarquia entre as classes e, por isso, as classes às quais os exemplos pertencem estão relacionadas entre si.

Enquanto a figura 3.2 ilustra as classes dos exemplos da tabela 3.2 e as suas relações, a figura 3.3 ilustra, através dos nós pintados com o fundo cinza, os nós usados para a construção de um classificador hierárquico a partir da base descrita na tabela 3.2. Os classificadores hierárquicos levam em consideração as classes associadas a todos os

nós da estrutura de classes. No caso da base descrita na tabela 3.2, seriam levadas em consideração as classes 1, 2, 1.1, 1.2, 2.1, 2.2 (Figura 3.3).

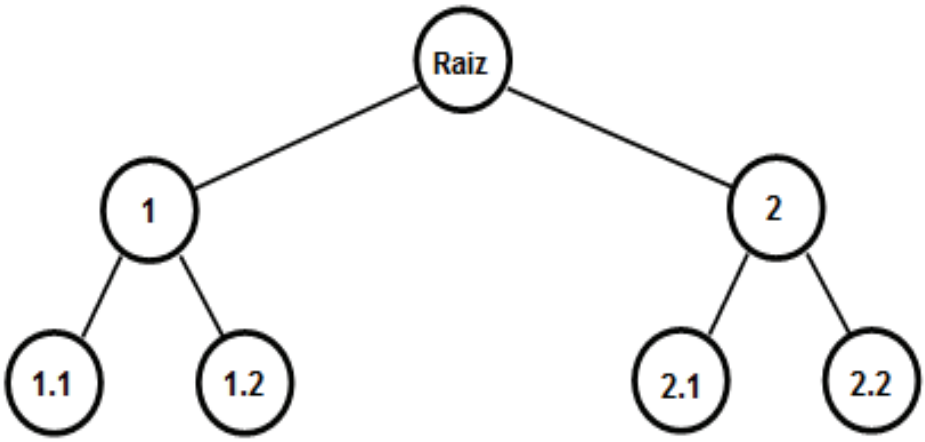

Figura 3.2 Árvore usada para representar as possíveis classes dos exemplos da base descrita na tabela 3.2.

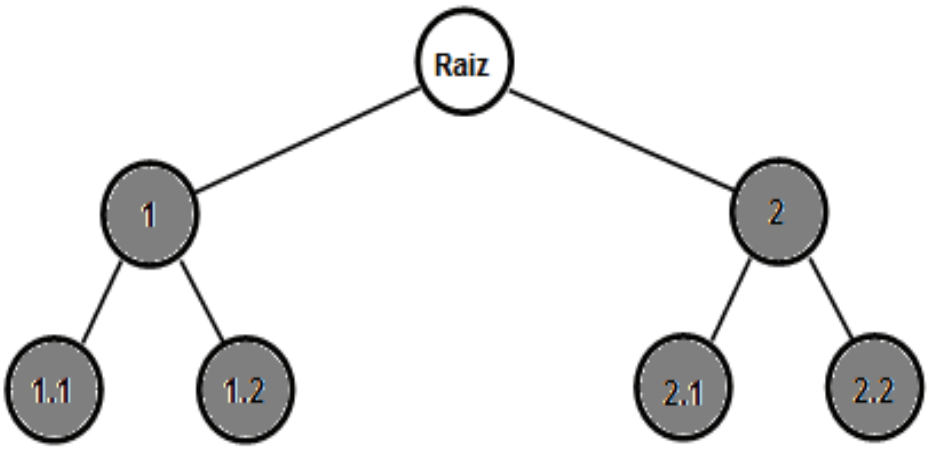

Figura 3.3 Árvore usada para representar as classes dos exemplos da base descrita na tabela 2.4 para a construção de um classificador hierárquico.

Na figura 3.3 estão pintados na cor cinza os nós associados às classes levadas em consideração na construção de um classificador hierárquico a partir da base descrita na tabela 3.2. Os classificadores hierárquicos podem ser usados em duas abordagens diferentes, sendo elas a local e a global [Silla and Freitas, 2011][Metz et al., 2011]. Caso uma base de dados seja composta por exemplos que podem pertencer a mais de

uma classe, o problema a ser resolvido é chamado multirrótulo. O classificador construído para esse tipo de base é chamado classificador multirrótulo. Caso a base seja composta por exemplos que podem pertencer a apenas uma classe, o problema a ser resolvido, assim como o classificador a ser construído, é chamado classificador de rótulo simples. Os classificadores locais são descritos na seção 3.1.1.1, sendo que todos exemplos usados nas seções seguintes vão abordar apenas problemas de rótulo simples, com cada explicação do tema sendo estendido para classificadores multirrótulo.

3.1.1.1 Classificadores hierárquicos locais

O classificador local pode ser construído segundo três abordagens: local por nível, local por nó e local por nó-pai. Enquanto o classificador local por nível tem um classificador por nível da estrutura usada para representação das classes, os classificadores por nó e por nó-pai têm classificadores associados aos nós [Silla and Freitas, 2011]. O classificador local por nó usa um classificador binário para cada nó (classe), sendo que é definido pelo classificador de cada nó se o exemplo pertence à classe associada a ele ou não [Silla and Freitas, 2011]. Como cada classificador é binário, os exemplos da base de dados devem ser tratados por ele como positivos ou negativos.

Existem diversas políticas de definição dos exemplos [Silla and Freitas, 2011]. A política mais usada é a que define que, para um classificador associado à classe *c*, os exemplos positivos são os que pertencem à própria classe *c*, além dos que pertencem às classes descendentes de *c* [Silla and Freitas, 2011]. Os exemplos negativos são aqueles pertencentes à classes irmãs de *c* e aos descendentes das classes irmãs de *c*.A seguir, será definido como é feita a escolha dos exemplos positivos e negativos para a construção do classificador por nó associado à classe 1.

Para a construção do classificador associado à classe 1, os exemplos pertencentes às classe 1 e às classes descendentes da classe 1 (1.1 e 1.2) são considerados positivos. Na figura 3.4 estão pintados com o fundo de cor cinza os nós associados às classes cujos exemplos pertencentes a elas são considerados positivos. Nesse caso, os exemplos 1, 2, 3 e 7 são considerados positivos, já que os exemplos 1, 2 e 7 pertencem a classes descendentes da classe 1 (1.2, 1.1, 1.2) e o exemplo 3 pertence à classe 1 (Tabela 3.3). Os exemplos 4 ,5 e 6, que não pertencem à classe 1 e nem às classes descendentes de 1, são considerados exemplos negativos.

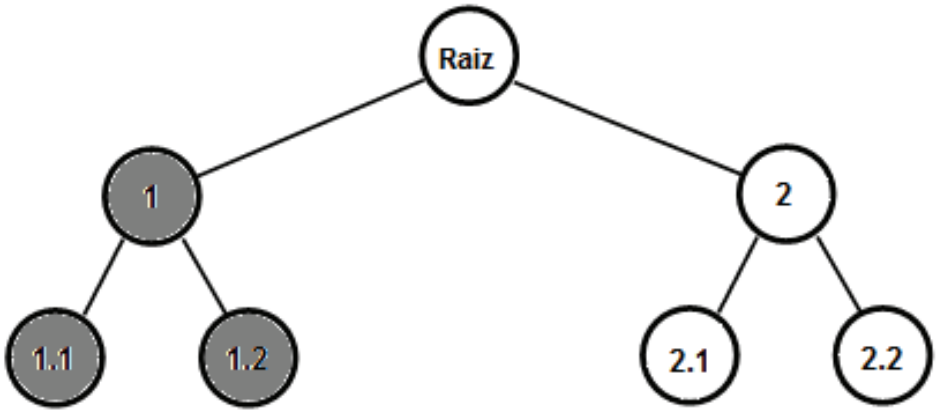

Figura 3.4 Árvore de ilustração das possíveis classes escolhidas pelo classificador associado à classe 1.

Tabela 3.3 Exemplo de uma base de dados com exemplos de proteínas usados para a construção do classificador local por nó associado à classe 1.

Exemplo/Atributo	Atributo A	Atributo B	Atributo C	Classe
1	9	5	8	Positivo
2	4	3	6	Positivo
3	3	1	2	Positivo
4	9	7	2	Negativo
5	5	6	3	Negativo
6	7	5	4	Negativo
7	9	8	1	Positivo

Seguindo a definição citada, no conjunto de treinamento do classificador local por nó associado ao nó 1, existem 4 exemplos Positivos (1, 2, 3 e 7) e 3 exemplos Negativos (4, 5, 6). Para a construção do classificador associado à classe 1.1 seriam considerados positivos os exemplos pertencentes à classe 1.1 e seus descendentes e negativos os exemplos pertencentes à classe 1.2 e seus descendentes. Os outros exemplos seriam descartados.

As regras que compõem o classificador por nó associado ao nó 1 (classe 1) podem ter como consequentes o valor “Positivo” ou “Negativo”. A seguir, será definido como é feita a escolha dos exemplos positivos e negativos para a construção do classificador por nó associado ao nó 2.1. Para a construção do classificador associado ao

nó 2.1 (classe 2.1), somente os exemplos pertences à classe 2.1 são classificados como positivos, já que a classe 2.1 não possui descendentes. Na figura 3.5 estão pintados com o fundo de cor cinza os nós associados às classes cujos exemplos pertencentes a elas são considerados positivos.

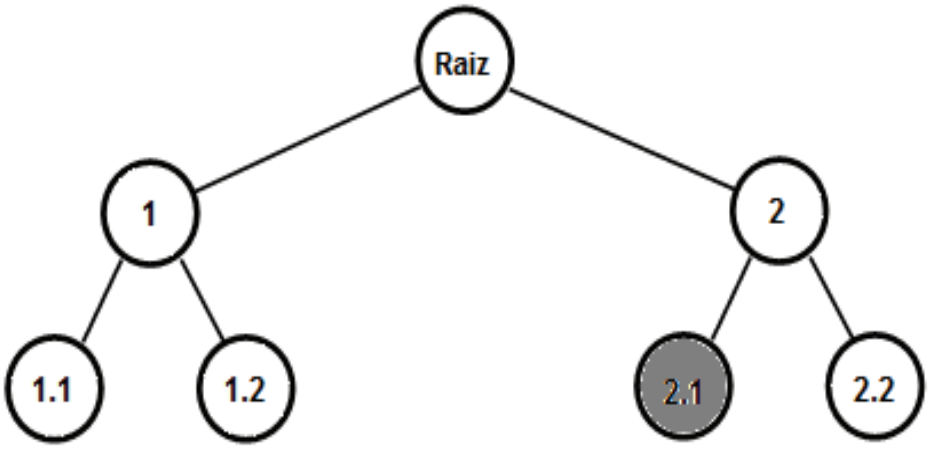

Figura 3.5 Árvore de ilustração das possíveis classes escolhidas pelo classificador associado à classe 2.1.

Como somente a classe 3.1 e seus descendentes são considerados positivos, para esse classificador, somente o exemplo 4 é considerado positivo, conforme descrito na tabela 3.4. Portanto, no conjunto de treinamento do classificador local por nó associado ao nó 2.1, existe um exemplo Positivo (4) e seis exemplos Negativos (1, 2, 3, 5, 6 e 7).

Tabela 3.4 Exemplo de uma base de dados com exemplos de proteínas usados para a construção do classificador local por nó associado à classe 1.

Exemplo/Atributo	Atributo A	Atributo B	Atributo C	Classe
1	9	5	8	Negativo
2	4	3	6	Negativo
3	3	1	2	Negativo
4	9	7	2	Positivo
5	5	6	3	Negativo
6	7	5	4	Negativo
7	9	8	1	Negativo

As regras que compõem o classificador por nó associado ao nó 2.1 podem ter como consequentes o valor “Positivo” ou “Negativo”. O classificador associado ao nó 1 pode definir que o exemplo pertence à classe 1 e o classificador associado ao nó 2.1 pode definir que ele pertence ao nó 2.1. Nesse caso, o classificador vai rotular o exemplo de forma inconsistente, já que um exemplo pertencente à classe 2.1 no nível 2 pertence à classe 2 no nível 1, sendo que o classificador definiu que o exemplo pertence à classe 1 no nível 1. A inconsistência do classificador por nó pode ser evitada com o uso da abordagem *top-down*. Usando essa abordagem, para a definição da classe à qual o exemplo pertence no nível *n* só são consideradas as classes descendentes da classe escolhida no nível *n*-1. Assim, para a construção de um classificador usando os dados descritos nas tabelas 3.3 e 3.4, caso o classificador associado ao nó 1 defina que o exemplo pertence à classe 1 no nível 1, apenas os classificadores associados às classes 1.1 e 1.2 seriam usados para a escolha da classe a qual o exemplo pertence no nível 2 [Silla and Freitas, 2011] [Costa et al., 2007].

No uso dos classificadores de rótulo simples e multirrótulo as regras são criadas com apenas um consequente. Para o classificador de rótulo simples, caso o classificador associado ao nó 1 defina que ele pertence à classe 1 e o classificador associado ao nó 2 defina que ele pertence à classe 2, a regra usada para definição associada ao classificador 1 deve ser comparada à regra usada para definição no classificador associado ao nó 2. A regra mais bem avaliada entre as duas terá a sua classe associada ao exemplo no nível 1. Para o classificador multirrótulo, para o caso citado, ao invés de comparar as regras usadas nos dois classificadores, as duas regras devem ser comparadas a um limiar [Bi and Kwok, 2010] sendo que todas as regras com medidas de avaliação maior do que o limiar têm a sua classe associada ao exemplo [Silla and Freitas, 2009].

O exemplo descrito para explicação do classificador local por nó tem as classes e as suas relações representadas por uma árvore. Em problemas nos quais as classes e suas relações são representados por um DAG esse tipo de classificador é usado da mesma maneira, sendo que cada nó tem um classificador binário associado a ele e somente os exemplos que pertencem à classe associada a ele e às classes associadas aos seus nós-filho são considerados positivos.

Na abordagem local por nó-pai é construído um classificador para cada nó não-folha da estrutura de classes. Assim, o classificador associado a um nó no nível *n* define

a qual classe o exemplo pertence no nível *n*+1. O classificador associado a um nó só pode definir a qual classe o exemplo pertence dentre aquelas associadas aos seus nós-filho [Silla and Freitas, 2011]. Usando o exemplos da base de dados descrita na tabela 3.4, nesta seção, inicialmente, será definido como é feita a escolha dos exemplos para a construção do classificador por nó associado ao nó Raiz. O nó Raiz é o nó-pai dos nós que pertencem ao primeiro nível da árvore de classes. Portanto, o classificador associado ao nó Raiz vai definir a qual classe o exemplo pertence entre as classes 1 e 2 (Figura 3.6).

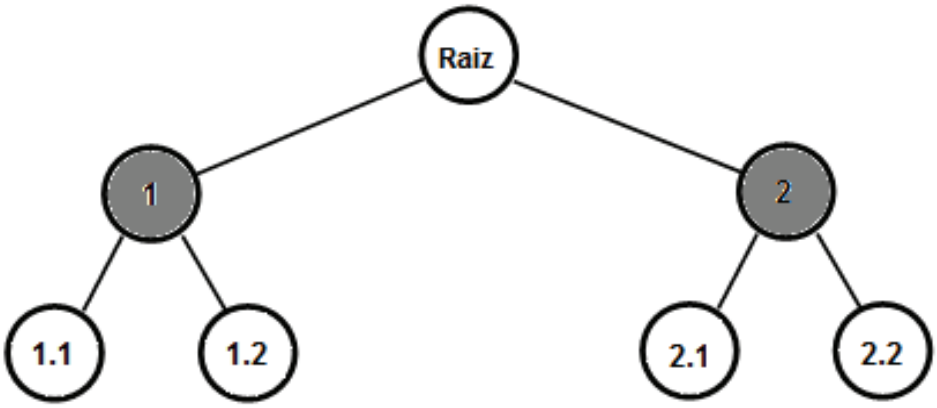

Figura 3.6 Árvore de ilustração de um classificador local por nó-pai associado ao nó-raiz.

Como um classificador por nó-pai associado a um nó é construído para se definir a qual das classes associadas aos seus nós-filhos o exemplo pertence, somente os exemplos pertencentes às classes associadas aos seus nós-filhos ou aos descendentes dos seus nós-filho são usados para compor o conjunto de treinamento. Os exemplos pertencentes às classes associadas aos descendentes dos seus nós-filhos são considerados como pertencentes à classe associada ao nó-filho em questão.

Os exemplos pertencentes às classes 1.1 e 1.2 são considerados como pertencentes à classe 1 e os exemplos pertencentes às classes 2.1 e 2.2 são considerados como pertencentes à classe 2. Portanto, o classificador associado ao nó Raiz usa os exemplos pertencentes aos seus nós-filhos (1 e 2) e aos seus descendentes (1.1, 1.2, 2.1 e 2.2) para definir a qual classe o exemplo está associada no nível seguinte (Tabela 3.5). Como o classificador vai definir a classe do exemplo entre as classes 1 e 2, as regras que compõem o classificador associado ao nó Raiz terão como consequentes a classe 1 ou a classe 2, com o mesmo procedimento sendo usado para a escolha da classe a qual o exemplo pertence no próximo nível da árvore.

Tabela 3.5 Exemplos pertencentes ao conjunto de treinamento do classificador associado ao nó Raiz.

Exemplo/Atributo	Atributo A	Atributo B	Atributo C	Classe
1	9	5	8	1
2	4	3	6	1
3	3	1	2	1
4	9	7	2	2
5	5	6	3	2
6	7	5	4	2
7	9	8	1	1

A seguir, será definido como é feita a escolha dos exemplos para a construção do classificador por nó associado ao nó 2, que vai definir a classe do exemplo dentre as classes descendentes da classe 2. Os exemplos que compõem o conjunto de treinamento do classificador associado ao nó 2 tem suas possíveis classes pintadas com o fundo de cor cinza na figura 3.6.

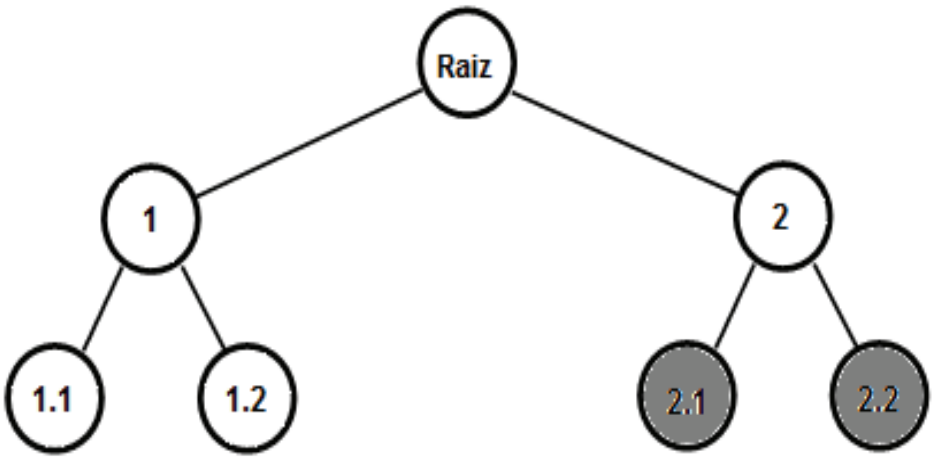

Figura 3.6 Árvore de ilustração de um classificador local por nó-pai associado à classe 2.

Como o classificador associado à classe 2 só pode prever as classes 2.1 e 2.2, (classes-filhas da classe 2), somente os exemplos pertencentes às classes 2.1 e 2.2 vão compor o conjunto de treinamento (Tabela 3.6). Além disso, como o classificador vai definir a classe do exemplo entre as classes 2.1 e 2.2, as regras geradas pelo classificador associado ao nó 2 terão como consequentes a classe 2.1 ou a classe 2.2.

Tabela 3.6 Exemplos usados para a construção do classificador associado à classe 2 com base nos exemplos da tabela 3.2 e da figura 3.6.

Exemplo/Atributo	Atributo A	Atributo B	Atributo C	Classe
4	9	7	2	2.1
5	5	6	3	2.2

As regras que compõem um classificador por nó-pai de rótulo simples possui apenas um consequente, sendo que o exemplo a ser rotulado vai pertencer à classe da regra mais bem avaliada (com maior grau de confiança) que o cobrir. Em um classificador multirrótulo, as regras têm apenas um consequente. Uma forma de se definir as classes do novo exemplo é usando um limiar para cada classificador, sendo que as regras que compõem o classificador e ultrapassarem esse limiar vão definir as classes do exemplo no nível em questão [Dumains and Chen, 2000].

O classificador por nó-pai e o classificador por nível geram menos classificadores do que os classificadores por nó. Enquanto o classificador por nó precisa de um classificador para cada nó, o classificador por nó-pai precisa somente de um classificador para cada nó não folha e o classificador por nível precisa de um classificador para cada nível. Assim, nos exemplos usados nas figuras de 2.2 a 2.8, o classificador local por nó precisa de classificadores para os nós 1, 2, 1.1, 1.2, 2.1 e 2.2. O classificador por nó-pai precisa de classificadores para os nós Raiz, 1 e 2. O classificador por nível precisa de classificadores para os níveis 1 e 2 [Silla and Freitas, 2011].

Os classificadores hierárquicos locais por nó, por nível (no uso da abordagem *top-down*) e por nó-pai, durante a fase de testes, têm a desvantagem de propagar um erro cometido em um dos níveis para os níveis seguintes. Assim, caso a estrutura de classes possua l níveis e a classe do nível n tenha sido escolhido de forma equivocada, as classes dos níveis n+1, n+2 até o nível l também serão escolhidos de forma equivocada, já que no nível n+1 somente as classes descendentes da classe escolhida no nível n são levadas em consideração, no nível n+2 somente as classes descendentes da classe escolhida no nível n+1 são levadas em consideração, e assim por diante [Silla and Freitas, 2011].

O classificador global não faz a classificação de um novo exemplo separada por níveis ou por nó e leva em consideração, no seu único classificador, toda a estrutura de classes (árvore ou DAG) para definir a classe do novo exemplo [Silla and Freitas, 2011] [Costa. et al, 2007]. No exemplo aqui usado, um classificador global seria construído para prever a classe do exemplo dentre todas as classes (1, 1.1, 1.2, 2, 2.1, 2.2). Usando os exemplos da base descrita na tabela 3.2, na figura 3.7 estão pintados com o fundo de cor cinza os nós associados às classes que o classificador global leva em consideração.

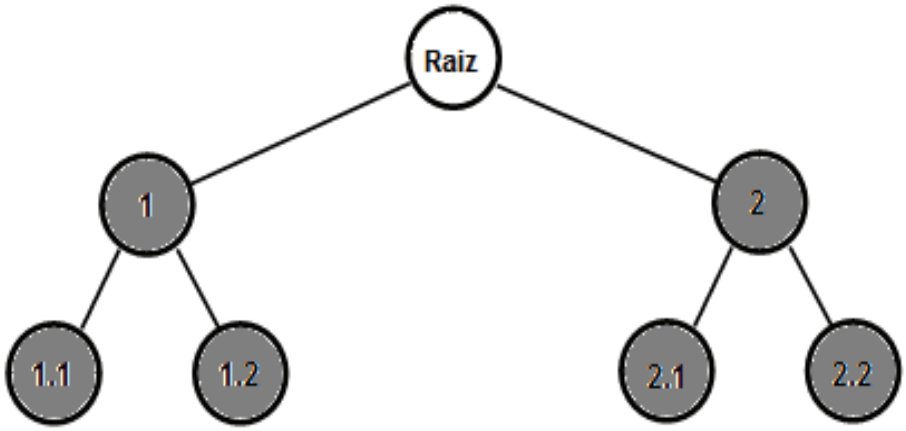

Figura 3.7 Grafo de ilustração de um classificador global

Como todas as classes são levadas em consideração, todos os exemplos da base compõem o conjunto de treinamento, como descrito na tabela 3.7.

Tabela 3.7 Exemplos usados para a construção de um classificador hierárquico global com base nos exemplos da tabela 3.2.

Exemplo/Atributo	Atributo A	Atributo B	Atributo C	Classe
1	9	5	8	1.2
2	4	3	6	1.1
3	3	1	2	1
4	9	7	2	2.1
5	5	6	3	2.2
6	7	5	4	2
7	9	8	1	1.2

As regras podem ter como consequente qualquer uma das classes possíveis para os exemplos da base de treinamento. Na abordagem de rótulo simples as regras que compõem o classificador global têm apenas um consequente. O novo exemplo a ser rotulado vai pertencer à classe da regra mais bem avaliada (com maior grau de confiança) que o cobrir.

A construção de um classificador global multirrótulo é feita com uma base de dados composta por exemplos pertencentes a uma ou mais classes. O modelo para um classificador multirrótulo pode ser criado com regras com um ou mais consequentes. Nesse caso, o número de consequentes de cada regra é um parâmetro a ser determinado pelo método usado para extração. O modelo para um classificador multirrótulo também pode ser composto por regras com apenas um consequente. Nesse caso, o método deve determinar o número de regras escolhidas para se rotular cada exemplo. Uma possível solução é o uso de um limiar pré-determinado, sendo que, para a classificação de novos exemplos, as classes previstas pelo classificador são as classes das regras com medidas de avaliação maior ou igual ao limiar determinado.

Como o classificador global abrange toda a estrutura de classes em um único conjunto de regras, a sua construção é mais complexa do que a construção dos classificadores locais [Costa et al., 2007]. Ainda não existem evidências claras de que o classificador hierárquico global seja mais eficiente do que o local [Silla and Freitas, 2011]. Assim, o motivo pelo qual o classificador hierárquico global merece ser usado é pelo fato de levar em consideração a hierarquia entre as classes de maneira natural [Silla and Freitas, 2009] e de gerar apenas um classificador, já que isso faz com que os modelos de classificação sejam mais claros e a tarefa de classificar objetos seja mais direta. Após a construção dos classificadores normalmente eles são avaliados, com o objetivo de se fazer a análise da viabilidade do seu uso por profissionais da área para auxílio a tomada de decisão. Alguns dos recursos que podem ser usados para se fazer a avaliação o de classificadores são descritos na seção 3.1.2.

3.1.2 Avaliação de classificadores

3.1.2.1 Avaliação de classificadores de rótulo simples com a estrutura de classes sendo representada por uma árvore

Para a avaliação de um classificador, os atributos e as classes de cada um dos exemplos da base de testes devem ser comparados com os antecedentes e com o

consequente de uma das regras que compõem o classificador. Dessa forma, para a avaliação de classificadores de rótulo simples a classe correta é a classe do exemplo a ser classificado e a classe prevista é a classe da regra escolhida pelo classificador para o exemplo.

Para a descrição das medidas de avaliação de classificadores com a estrutura de classes sendo representada por árvores, serão definidos os conceitos de Verdadeiro Positivo, Verdadeiro Negativo, Falso Positivo e Falso Negativo, usados para classificadores binários. Assim, para cada exemplo da base de testes:

Verdadeiro Positivo(VP) –Classe prevista para o exemplo = "Positivo" e Classe correta do exemplo *x* = "Positivo".

Verdadeiro Negativo(VN) –Classe prevista para o exemplo = "Negativo" e Classe correta do exemplo *x* = "Negativo".

Falso Positivo(FP) –Classe prevista para o exemplo = "Positivo" e Classe correta do exemplo *x* = "Negativo".

Falso Negativo(FN) –Classe prevista para o exemplo = "Negativo" e Classe correta do exemplo *x* = "Positivo".

NVP é o número de VPs ocorridos na classificação de todos os exemplos da base de testes, NFP o número de FPs e NFN o número de FNs. A partir disso podem ser definidas as medidas Precisão (*P*), Revocação (*R*) e a medida *F*. Essas medidas são descritas nas equações 3.1, 3.2 e 3.3 (Olson and Delen, 2008).

$$\boldsymbol{P = NVP/(NVP + NFP)} \tag{3.1}$$

$$\boldsymbol{R = NVP/(NVP + NFN)} \tag{3.2}$$

$$\boldsymbol{F = ((T^2 + 1) * P * R)/(T^2 * P + R)} \tag{3.3}$$

O valor de *T* geralmente é 1.

[Kiritchencko et al., 2005] propuseram três medidas para uso em classificadores hierárquicos. As medidas são chamadas de precisão hierárquica (hP), revocação hierárquica (hR) e medida hF.

A precisão hierárquica é calculada como descrito na equação 3.4.

$$\boldsymbol{hP} = \frac{\sum_{x=1}^{P} \boldsymbol{hP_x}}{\boldsymbol{P}} \quad (3.4)$$

P é o número de exemplos da base de testes e *hPx* é descrita na equação 3.5.

Para o cálculo da medida hP_x serão definidos dois conjuntos, sendo eles *C* e *Cx*. *C* é composto pela classe prevista e seus ancestrais, enquanto *Cx* é composto pela classe correta do exemplo *x* e seus ancestrais.

$$\boldsymbol{hP_x} = \boldsymbol{A_x/B} \quad (3.5)$$

Ax é o número de elementos em comum entre *C* e *Cx* e *B* é o número de elementos presentes em *C*.

A revocação hierárquica é calculada como descrito na equação 3.6.

$$\boldsymbol{hR} = \frac{\sum_{x=1}^{P} \boldsymbol{hR_x}}{\boldsymbol{P}} \quad (3.6)$$

hR_x é descrita na equação 3.7.

Para um determinado exemplo a medidas *hR* do classificador é calculada como na equação 3.7:

$$\boldsymbol{hR_x} = \boldsymbol{A_x/E} \quad (3.7)$$

E o número de elementos presentes em *Cx*.

A medida hF de uma regra é calculada como descrito na equação 3.8.

$$\boldsymbol{hF} = ((\mathbf{T^2} + \mathbf{1}) * \boldsymbol{hP} * \boldsymbol{hR})/(\mathbf{T^2} * \boldsymbol{hP} + \boldsymbol{hR}) \quad (3.8)$$

Devido às diferenças existentes na estrutura das árvores e dos DAG´s, a avaliação dos classificadores também é feita de maneira diferente. A forma como pode ser feita a avaliação de classificadores de rótulo simples com a estrutura de classes representada por um DAG é descrita na seção 3.1.2.2.

3.1.2.2 Avaliação de classificadores de rótulo simples com a estrutura de classes sendo representada por um DAG

A avaliação de classificadores de rótulo simples com a estrutura de classes representada por um DAG pode ser feita usando-se as mesmas medidas usadas na avaliação dos classificadores de rótulo simples com a estrutura de classes representada por uma árvore. Com o uso de um DAG, as medidas de precisão e revocação

hierárquicas do classificador também são calculadas por meio da precisão (*hPx*) e revocação (*hRx*) hierárquicas para cada exemplo da base.

Porém, as equações para o cálculo de hP_x e hR_x devem ser modificadas. Para isso, considera-se o cálculo da distância entre duas classes, sendo elas *cp* (classe prevista pelo classificador para o exemplo) e *cc* (classe do exemplo). A classe *cp* é considerada um sub-grafo, chamado aqui de *SGCP* e composto por todos os caminhos que ligam o nó Raiz da DAG ao nó que representa *cp*. O conjunto de todas as sub-árvores que compõem SGCP é chamado aqui de *SACP*. Além disso, a classe *cc* é um sub-grafo, chamado aqui de *SGCC* e composto por todos os caminhos que ligam o nó Raiz ao nó que representa a classe *cc.* O conjunto de todas as sub-árvores que compõem SGCC é chamado aqui de *SACC*.

O cálculo da distância entre *cp* e *cc* tem como resultado *V(cp,cc),* que representa o valor da distância entre *cp* e *cc*, e *saSACP* e *saSACC*, que representam, respectivamente, as sub-árvores dos conjuntos *SACP* e *SACC,* usadas para o cálculo de *V(cp,cc)*. Assim, será definida uma matriz *Q* de dimensão *a* x *b*, sendo a o número de elementos de *SGCP* e b o número de elementos de *SGCC*. *Q* é calculada de acordo com a equação 3.9.

$$\boldsymbol{Q_{ij} = d(SACP_i, SACC_j)} \tag{3.9}$$

Em que *SACPi* é o *i*-ésimo elemento do conjunto *SACP*, $SACC_j$ é o *j*-ésimo elemento do conjunto *SACC*. A distância $d(SACP_i,SACC_j)$ é calculada como descrito na equação 3.10.

$$\boldsymbol{d(SACP_i, SACC_j) = dSACP_i + dSACC_j} \tag{3.10}$$

Em que $dSACP_i$ (equação 3.11) é a distância entre o nó-folha da sub-árvore $SACP_i$ e o nó de nível *NCom*, $\mathrm{dSACC_j}$ (equação 3.12) a distância entre o nó-folha da sub-árvore $SACC_j$ e o nó de nível *NCom*. *NCom* é o número de nós em comum entre $SACP_i$ e $SACC_j$.

$$\boldsymbol{dSACP_i = \sum_{i=NCom+1}^{m} p_i} \tag{3.11}$$

Em que *p* é o peso do nível *i* e *m* o número de elementos de *SACPi*.

$$\boldsymbol{dSACC_j = \sum_{i=NCom+1}^{z} p_i} \tag{3.12}$$

Em que *z* é o número de nós de *SACCj*.

Um dos problemas do cálculo da distância entre duas sub-árvores é que erros cometidos em diferentes níveis são tolerados da mesma maneira. Usando as classes descritas nas figuras 3.2 a 3.8, por exemplo, caso as classes previstas pelo classificador sejam as classes 1 e 2.1 e as classes corretas sejam as classes 2 e 2.2, a distância entre as classes prevista e correta, nos dois casos, seriam de duas arestas. Dessa forma, nos dois casos, o classificador seria punido com a mesma intensidade. Porém, como a predição de classes mais específicas são mais complexas de serem feitas, da classe 1.2.1.1, um erro cometido pelo classificador em níveis mais específicos deve ser mais tolerado do que erros cometidos em níveis menos específicos [Costa et al., 2007] [Holden and Freitas, 2008b]. Por isso, são usados pesos para os níveis nas equações 2.11 e 2.12, como proposto por [Blockeel et al., 2002].

A distância *d(SACPi,SACCj)* é usada como parte do cálculo da distância entre dois sub-grafos, mas também pode ser usada, de maneira isolada, para o cálculo da distância entre duas sub-árvores. Dessa forma, o peso para cada nível é usado tanto para o cálculo da distância entre dois sub-grafos quanto para o cálculo da distância entre duas sub-árvores. O valor de *V(cp,cc)* é calculado como descrito na equação 3.13.

$$\mathbf{V(cp, cc) = val(M)} \tag{3.13}$$

Em que *val* é o valor da célula *M*, definida como descrito na equação 3.14.

$$\mathbf{M = maxcel(Q)} \tag{3.14}$$

Em que *maxcel(Q)* é a célula de maior valor da matriz *Q*.

A sub-árvore *saSACP* é definida como descrito na equação 3.15.

$$\mathbf{saSACP = Am} \tag{3.15}$$

Em que *m* é a linha da matriz onde *M* está situada.

A sub-árvore *saSACC* é definida como descrito na equação 3.16

$$\mathbf{saSACC = Bn} \tag{3.16}$$

Em que *n* é a coluna da matriz onde *M* está situada.

Para a avaliação de classificadores hierárquicos de rótulo simples usando DAG, o cálculo de *hPx* e *hRx* (equações 3.5 e 3.7, respectivamente), *C* é composto pelos nós da sub-árvore *saSACP* e *Cx* é composto pelos nós da sub-árvore *saSACC*.

Devido à diferença entre o número de classes que um exemplo pode pertencer nos classificadores de rótulo simples e multirrótulo, esses dois tipos de classificadores são avaliados de maneiras distintas. Na seção 3.1.3.3 e 3.1.3.4 são descritas algumas das formas de avaliação de classificadores multirrótulo com a hierarquia entre as classes representada por uma árvore e por um DAG, respectivamente.

3.1.3.3 Avaliação de classificadores de multirrótulo com a estrutura de classes sendo representada por uma árvore

Os classificadores multirrótulo são construídos a partir de bases de dados com exemplos que pertencem a uma ou mais classes. Assim, o novo a exemplo a ser classificado pode pertencer a uma ou mais classes. O classificador pode ser composto por regras com mais de um consequente, sendo que, durante a fase de testes, no momento da classificação, será escolhida uma regra pelo classificador para rotular o novo exemplo. Uma alternativa é a criação de regras com apenas um consequente, sendo que, no momento da classificação, as regras com medidas de avaliação acima de um limiar determinado serão escolhidas para classificação de cada exemplo da base.

As medidas *hR*, *hP* e *hF* [Borges, 2012] [Romão, 2012] têm sido usadas para a avaliação de classificadores hierárquicos multirrótulo. Para a classificação de um exemplo *x*, o conjunto *Corretas* é composto por todas as *ncp* classes previstas pelo classificador para o exemplo *x*. O conjunto *Previstas* é composto por todas as *nc* classes do exemplo *x* pertence. A partir daí será formado o grupo *Pares*, composto por *nc* pares. Para a formação do conjunto *Pares* é construída uma matriz quadrada *MD* de dimensão*ncp* x *nc*. Cada elemento de índice *ij* é calculado como descrito na equação 3.17.

$$\mathbf{MD_{ij}} = \mathbf{d}\left(\mathbf{Corretas_i}, \mathbf{Previstas_j}\right) \tag{3.17}$$

Em que *d*(*Corretas i*,*Previstas j*) é a distância entre a classe i do conjunto *Corretas* e *j* do conjunto *Previstas*. Considerando cada uma das classes uma sub-árvore, *d*(*Corretas i*,*Previstas j*) pode ser calculada como descrito na equação 3.10.

O par *i* do conjunto *Pares* é formado pelo elemento *i* do conjunto *Corretas* e o elemento *mi* do conjunto *Previstas*, sendo *mi* a coluna do elemento de menor valor da linha *i*. Após a formação do par *i*, todos os elementos da coluna *mi* devem ter valor 1.1. Isso é feito para que seja anulada a possibilidade de qualquer um dos elementos dos

conjuntos *Previstas ou Corretas* estar contido em dois elementos do conjunto *Pares*, já que o valor máximo para *d*(*Corretas i*,*Previstas j*) é 1.

Assim, *hRx* e *hPx* são calculadas por meio das equações 3.18 e 3.19.

$$\boldsymbol{hP_x = (\sum_{y=1}^{nc}(Co/\mathbf{Nec}))/\mathbf{nc}} \quad (3.18)$$

Em que *Co* é o número de elementos em comum entre *Epr* e *Ec, Nec* é o número de elementos presentes em *Ec*, *Epr* é composto pelos nós da sub-árvore do primeiro elemento do par *y* do conjunto *Pares* e *Ec* é composto pelos nós da sub-árvore do segundo elemento do par y do conjunto *Pares*.

$$\boldsymbol{hR_x = (\sum_{y=1}^{j}(Co/\mathbf{R}))/\mathbf{j}} \quad (3.19)$$

Em que *R* é o número de elementos presentes em *Epr*.

A medida AU (PRC) (do inglês *Area Under Precision-Recall Curve*) é usada para a avaliação de classificadores hierárquicos multirrótulo [Vens et al., 2008]. Para o cálculo da AU (PRC) deve ser construída a curva PR (ou PRC, do inglês *Precision-Recall Curve*), que também têm sido usada para a avaliação de classificadores hierárquicos multirrótulo [Vens et al., 2008] [Otero et al, 2010] [Borges, 2012] [Romão, 2012]. Para a construção da curva PR são usadas a precisão e a revocação não hierárquicas. Como essas medidas são usadas para problemas binários, deve ser usado um limiar para que elas sejam usadas em problemas não binários. Além disso, cada regra que compõe o classificador deve ter um grau de confiança [Vens et al., 2008]. Assim, dado um limiar *l*, para cada exemplo *x* pertencente à classe *c2*, uma regra prevista pelo classificador para o exemplo *x* (r) com consequente (classe) *c1* e grau de confiança *g*, tem-se:

Verdadeiro Positivo(VP) – Se (g>=l) e (c1=c2).

Falso Positivo(FP) – Se (g>=l) e (c1<>c2).

Falso Negativo(FN) – Se (g<l) e (c1=c2).

Os valores de *NVP*, *NFP*, *NFN, P e R* são calculados como descrito na seção 3.2.2.1. Cada ponto da curva *PR* é construído com o uso de um limiar diferente, sendo que os limiares variam de 1 a 0. A curva PR é construída com a revocação no eixo *x* e a precisão no eixo *y*. Para o cálculo da medida AU(PRC) é calculada a área sob a curva PR.

Os classificadores multirrótulo com a estrutura de classes sendo representada por um DAG são avaliados de maneira diferente daqueles que tem a estrutura representada por uma árvore. Isso ocorre devido à maior complexidade presente no primeiro tipo de classificador citado, que tem algumas de suas formas de avaliação descritas na seção 3.1.3.4.

3.1.3.4 Avaliação de classificadores multirrótulo com a estrutura de classes sendo representada por um DAG

A avaliação de um classificador com a estrutura de classes representada por um DAG pode ser feita da mesma forma como para um classificador com a estrutura de classes representada por uma árvore. Portanto, para a classificação de um exemplo *x*, o conjunto *Previstas* é composto pelas *ncp* classes previstas pelo classificador para o exemplo *x*. O conjunto *Corretas* é composto pelas *nc* classes do exemplo x. A partir daí, será formado o grupo *Pares*, composto por *nc* pares. Porém, para a formação dos elementos de *Pares* são formadas três matrizes quadradas de dimensão *nc*, sendo elas *MCD (Matriz de Cálculo das Distâncias)*, *MCC* (Matriz de classes Corretas), *MCP* (Matriz de classes previstas).

Para o cálculo dos valores das células das matrizes MCD, MDD e MCP serão usados os valores obtidos por meio do cálculo da distância entre dois sub-grafos que representam a classe prevista e a classe correta (o valor *V(cp,cc)* e as sub-árvores *saSACP* e *saSACC*), definido na seção 3.2.2.1. Para a célula *ij* da matriz MCD, tem-se o valor *V(cp,cc)* obtido do cálculo da distância entre a classe *i* do conjunto *Previstas* e a classe *j* do conjunto *Corretas*. Para a célula *ij* da matriz MCC, tem-se a sub-árvore *saSACP* obtida do cálculo da distância entre a classe *i* do elemento *Previstas* (*sgSACP*) e a classe *j* do elemento *Corretas*. Para a célula *ij* da matriz MCP, tem-se a sub-árvore *saSACC* obtida do cálculo da distância entre a classe*i* do elemento *Previstas* (*sgSACP*) e a classe *j* do elemento *Corretas* (*sgSACC*).

O elemento *i* do conjunto *P* é formado pelo elemento *nij* da matriz *MCC* e *nij* da matriz *MCP*, sendo *nij* a coluna do elemento de menor valor da linha *i* da matriz *MCD*. Para que seja anulada a possibilidade de um elemento do conjunto *U* estar contido em dois elementos do conjunto *Pares*, todos os elementos da coluna *i* das três matrizes passam a ser desconsiderados. A partir desse ponto o procedimento é o mesmo usado quando as classes são representadas por uma árvore.

Para o cálculo da curva PR, como não é levada em conta a hierarquia, as medidas de precisão e revocação para um classificador multirrótulo com a hierarquia de classes representada por um DAG são calculadas da mesma forma como são calculadas para os classificadores com a estrutura de classes representada por uma árvore.

Na seção 3.1 foram descritos os classificadores e suas formas de avaliação. Neste trabalho, o algoritmo usado para a construção dos classificadores é baseado na ED, descrita na seção 3.2.

3.2 Evolução Diferencial (ED)

A Evolução Diferencial (ED) é um modelo matemático pequeno e simples do processo de evolução [Feoktistov, 2006] criado para a resolução de problemas de otimização em espaços de estados com atributos contínuos [Storn and Price, 1997].

3.2.1 Estrutura

A ED é formada por uma população de *N* indivíduos ou cromossomos, que evoluem e passam por *G* gerações. Cada indivíduo (vetor) *I* representa uma possível solução do problema a ser resolvido. Essas soluções (indivíduos) são avaliadas levando-se em consideração a sua proximidade em relação à estimada solução do problema. A pontuação do indivíduo *I* advinda da avaliação é chamada de *fitness* (*FI*), que também pode ser definido como critério de otimização [Feoktistov, 2006].

Um indivíduo *I* é formado por *e* genes, sendo que cada gene representa um elemento da solução candidata [Feoktistov, 2006] [Storn and Price, 1995]. Os valores de *N* e *G* dependem do problema que está sendo solucionado. Alguns autores sugerem que o valor de *N* seja calculado a partir do valor de *e* [Das et al., 2009] [Gamperle et al., 2002] e que o valor de *G* seja calculado a partir do valor de *N* [Brest and Maucec, 2008]. Os elementos descritos nesta seção funcionam em conjunto e como descrito na seção 3.2.2.

3.2.2 Funcionamento

Os indivíduos da primeira geração (g=1) são criados, dentro do domínio do problema, aleatoriamente. Para uma geração g (1<=g<G), para cada indivíduo *I*, após a sua avaliação, um novo indivíduo *NI* é formado a partir dos mecanismos de mutação e cruzamento (Figura 3.11). Posteriormente, para que haja evolução na passagem de uma

geração para outra, o indivíduo mais apto entre *I* e *NI* sobrevive para a próxima geração (Figura 3.8). A esse processo é dado o nome de seleção.

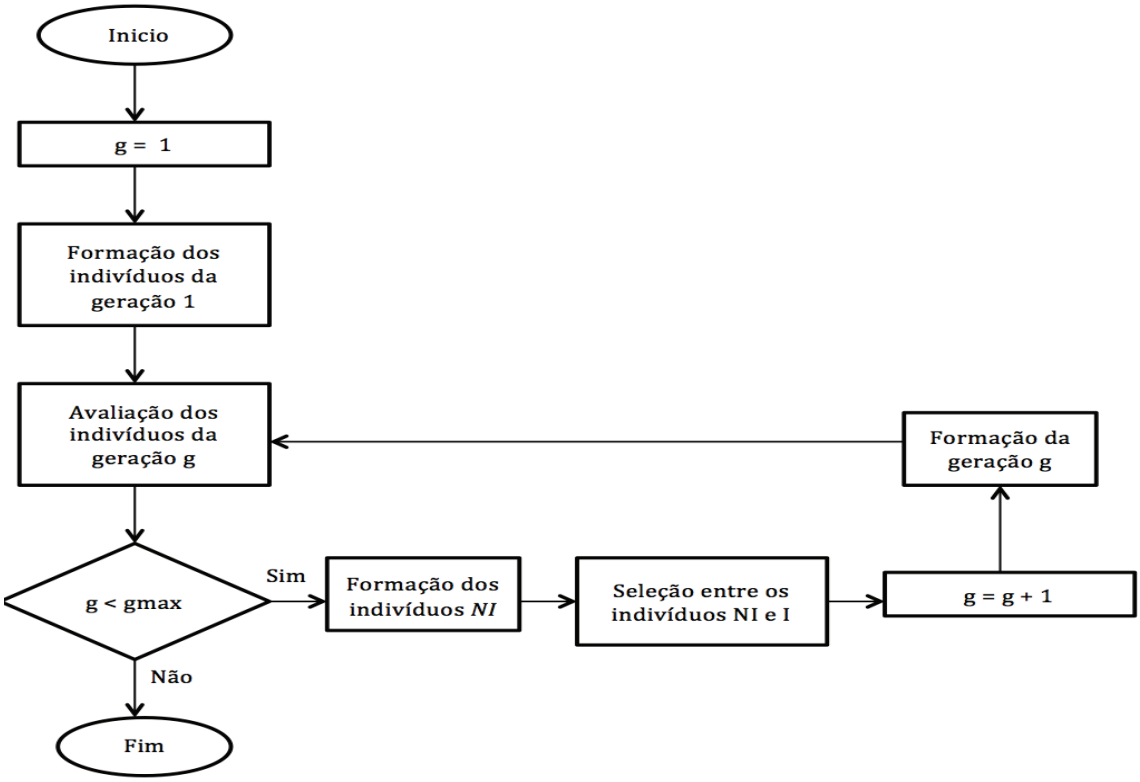

Figura 3.8 Fluxograma da ED.

Como é ilustrado no fluxograma da figura 3.8, os procedimentos de mutação, cruzamento e seleção são repetidos até que a geração *G* seja formada.

3.2.3 Formação dos novos indivíduos (*NI*)

Cada indivíduo *NI* é formado usando os procedimentos de mutação e cruzamento. Inicialmente, realizando a mutação, é formado um indivíduo chamado aqui de *TI*. Posteriormente, a partir do cruzamento entre *I* e *TI* é formado o indivíduo *NI*. A mutação para formação de *TI* é descrita na seção 3.3.3.1, o cruzamento para formação de *NI* na seção 3.3.3.2 e a seleção para a escolha dos indivíduos que sobrevivem para a geração seguinte na seção 3.3.4.

3.2.3.1 Mutação

O indivíduo *TI* é formado a partir da mutação de outros indivíduos da população. Assim, para cada vetor TI de uma geração *g* um novo vetor *TI* é formado a partir de outros vetores da mesma geração (Figura 3.9).

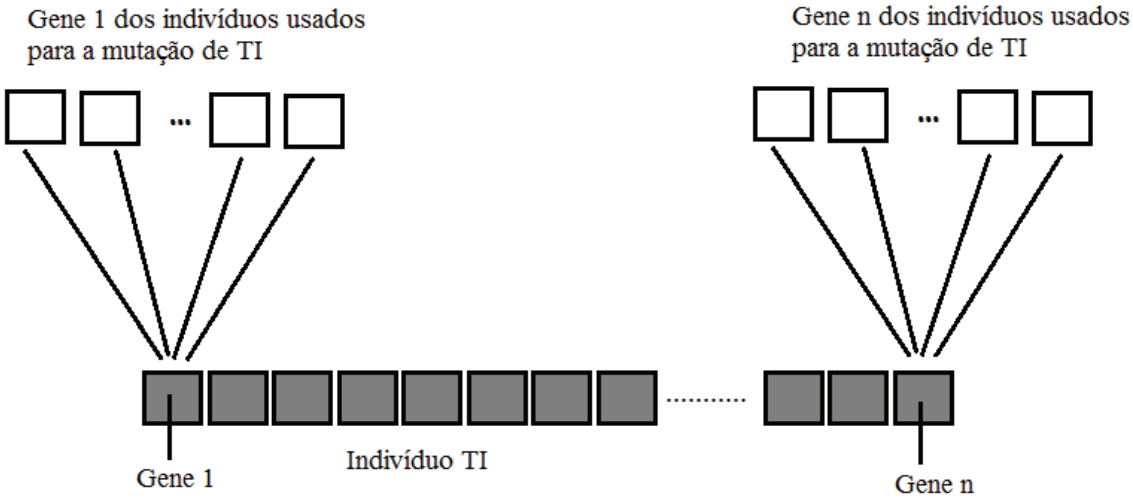

Figura 3.9 Representação do mecanismo de mutação.

Na figura 3.9, os genes dos indivíduos *TI* estão marcado com o fundo de cor cinza, enquanto os genes dos indivíduos usados para a mutação do indivíduo *TI* estão com o fundo de cor branca.

A escolha dos indivíduos usados para a mutação de *TI* depende da estratégia usada [Feoktistov, 2006]. A notação usada para definir o nome de cada estratégia tem o objetivo de classificar as diferentes estratégias da ED. Assim, cada estratégia é denominada DE/a/b/c, sendo *a* a forma de escolha do indivíduo a ser modificado (podendo ser, por exemplo, *rand* caso o vetor seja escolhido de maneira randômica ou *best* caso seja escolhido o indivíduo mais bem avaliado da população) , *y* o número de diferenças entre vetores usadas para o cálculo da mutação e *z* a estratégia de cruzamento usada [Storn, 1996] [Das et al., 2009].

Uma dessas estratégias foi proposta por [Storn and Price, 1995] e é descrita por meio da equação 3.20.

$$\boldsymbol{TI = S1 + FM * (S2 - S3)} \tag{3.20}$$

Em que *S1* é o primeiro indivíduo sorteado, *FM* o fator de mutação, *S2* é o segundo indivíduo sorteado e *S3* é o terceiro indivíduo sorteado. [Feoktistov, 2006] sugere FM = 0.4. A estratégia descrita pode ser referenciada como DE/rand/1, já que o indivíduo da população usado como base para mutação é escolhido aleatoriamente (S1) e há apenas uma diferença calculada entre os indivíduos (S2-S3). A variável que define o tipo de cruzamento usado não consta na definição porque essa notação define apenas a estratégia de mutação usada.

Foi proposta também por [Storn and Price, 1995] uma estratégia chamada DE/target-to-best/1. descrita por meio da equação 3.21.

$$\boldsymbol{TI = I + FM * (M - I) + FM * (S1 - S2)} \quad (3.21)$$

Em que *M* é o indivíduo mais bem avaliado da população. O valor da variável *b* na estratégia citada acima é 1 porque é considerado que o vetor usado para mutação é uma recombinação, resultado da equação I +FM*(M+I).

A estratégia a ser usada depende do problema que está sendo resolvido. A estratégia DE/rand/1, por exemplo, privilegia o preenchimento de todo o espaço de estados já que nela os indivíduos usados para a mutação de *TI* são escolhidos aleatoriamente, sem levar em consideração o *fitness* de nenhum deles. Dessa forma, os indivíduos *NI* são gerados de maneira aleatória, sem levar em consideração o indivíduo *I* e os indivíduos mais bem avaliados da população. Isso faz com que não haja evolução do indivíduo *I*. Além disso, por não haver critério para definição dos indivíduos a serem usados na mutação, o algoritmo pode demorar para encontrar a solução ou estagnar em uma região do espaço de estados onde não esteja a solução ótima [Veterstrom and Thomsen, 2004]. Normalmente, esse tipo de estratégia é usada em problemas nos quais se deseja encontrar um conjunto de soluções heterogêneas, e não somente uma solução. Um exemplo desse tipo de problema, descrito a seguir, é o cálculo de valores para taxas de 4 substâncias com o objetivo de alcançar um determinado valor da taxa de insulina. A taxa de insulina no sangue é influenciada pelo valor das taxas de outras substâncias, como por exemplo a glicose, o glucagon, o glicogênio e o cortisol. Hipoteticamente, a relação entre essas quatro substâncias pode ser representada pela equação 3.22:

$$\mathbf{2X + 3Y + 4Z - 7W = i} \quad (3.22)$$

As variáveis *X*, *Y*, *Z* e *W* representam as taxas de glicose, glucagon, glicogênio e cortisol, respectivamente, enquanto a variável *i* representa a taxa de insulina. A ED pode ser usada para se encontrar uma solução para essa equação (valores para X, Y, Z e W que, juntos, satisfaçam a equação). Porém, o ideal é que sejam encontradas mais de uma solução, para que o médico tenha opção de mudar a dosagem dos medicamentos de acordo com o comportamento que o organismo do paciente apresente ao longo do tratamento. Como a estratégia DE/rand/1 privilegia a abrangência do espaço de estados durante toda a execução do algoritmo, o seu uso aumenta as chances de serem encontradas soluções em diferentes regiões do espaço de estados.

A estratégia DE/target-to-best/1 privilegia a formação de *NI* a partir da evolução do indivíduo *I* usando as características do indivíduo *M*. Ao contrário da estratégia DE/rand/1, a estratégia DE/target-to-best/1 escolhe os indivíduos para a mutação com base no *fitness*. Esse tipo de estratégia é usada quando se deseja encontrar a solução ótima.

A estratégia DE/target-to-best também pode ser usada para a resolução do problema relacionado à equação 3.22. Porém, como os indivíduos sofrem mutação tentando imitar o indivíduo *M*, a tendência é que, com o passar das gerações, todos os indivíduos fiquem com as características mais parecidas com a dele. Assim, o algoritmo pode convergir para uma região do espaço de estados em que existe um máximo local [Buhry et al., 2009][Das et al., 2009][Zhang and Sanderson, 2009], fazendo com que o algoritmo não encontre a solução ótima.

Portanto, caso se deseje encontrar soluções heterogêneas, as estratégias que privilegiam a exploração do espaço de estados, como a DE/rand/1, são mais vantajosas por percorrer de maneira mais abrangente o espaço de estados. Caso se deseje encontrar apenas uma solução, as estratégias que privilegiam a evolução, como a DE/target-to-best, são mais vantajosas, por evoluir os indivíduos *I*, imitar o indivíduo *M* e por isso encontrar uma solução mais rapidamente. Após a mutação, os indivíduos sofrem cruzamento, que funciona da maneira descrita na seção 3.2.3.2.

3.2.3.2 Cruzamento

Após a formação de *TI*, independente da estratégia usada, os indivíduos *I* e *TI* sofrem cruzamento para a formação do indivíduo NI (Figura 3.10).

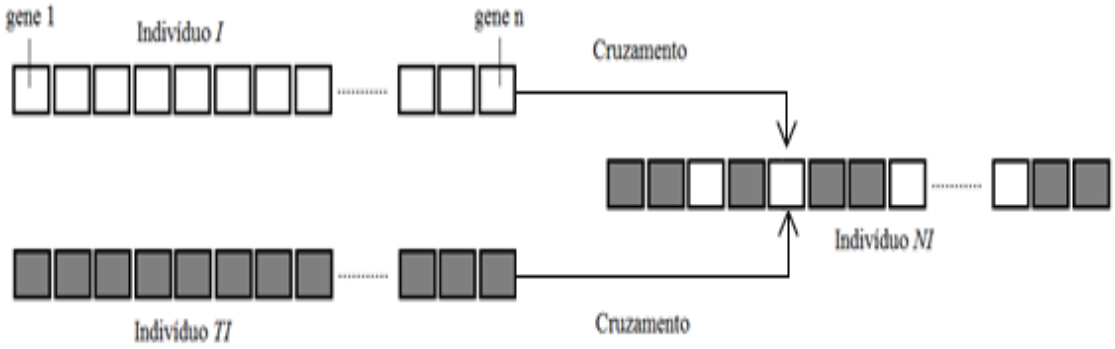

Figura 3.10 Ilustração do mecanismo de cruzamento.

Existem, principalmente, dois tipos de cruzamento, sendo eles o binomial e o exponencial [Zaharie, 2007]. Apesar de o exponencial ter sido proposto no mesmo trabalho em que a ED foi proposta [Storn and Price, 2005], o binomial é mais comumente usado [Zaharie, 2007]. No cruzamento binomial, para a formação de cada gene *i* de *NI*, um número, chamado aqui de *rand(i)*, é escolhido aleatoriamente(0<=*rand(i)*<=1). Além disso, é usada uma taxa de cruzamento, chamada aqui de *Tc*. Assim:

NIi = *TIi*, caso *rand(i)*<=*Tc*.

NIi = *Ii*, caso contrário.

Em que *NIi* é o gene *i* do indivíduo *NI* e *TIi* o gene *i* do indivíduo T*I*. Após o cruzamento, os indivíduos são selecionados como descrito na seção 3.2.4 para a composição da geração seguinte.

3.2.4 Seleção

Após a formação de *NI*, a partir do processo de seleção, um dos indivíduos entre *I* e *NI* é escolhido para compor a geração *g*+1. Portanto:

I(g+1) = *I(g)*, caso *FI*>=*FNI*,

I(g+1) = *NI(g)*, caso contrário.

Como todos os componentes da ED foram descritos, na seção 3.2.5 será mostrado um exemplo de funcionamento do algoritmo.

3.2.5 Exemplificação do funcionamento da ED

Nesta seção será descrita parte da resolução do problema descrito na seção 3.3.3.1, cuja solução é encontrar valores para as taxas de 4 substâncias com base no valor da taxa de insulina. Assim, o valor desejado de insulina deve ser determinado para que, por meio da ED sejam encontrados os valores das taxas das 4 substâncias.

Para o exemplificar a técnica, será usado o valor de i = 15.7. Nesse exemplo serão usados 5 indivíduos para tornar o exemplo mais simples. Em uma aplicação real, como sugerido por [Das et al., 2009], o algoritmo deveria ter 40 (10*e) indivíduos.

Como, nesse caso, uma solução deve ter 4 valores contínuos (para X, Y, Z e W), cada indivíduo nesse exemplo tem 4 genes. Como os indivíduos da primeira geração

são gerados aleatoriamente dentro do domínio do problema, nesse exemplo a primeira geração é composta pelos indivíduos descritos na tabela 3.8. Para os indivíduos apresentados na tabela 3.8 o gene 1 representa o valor de X, o gene 2 representa o valor de Y, o gene 3 representa o valor de Z e o gene 4 representa o valor de W.

Tabela 3.8 Indivíduos da 1a geração da exemplificação da ED.

Indivíduo/Atributo	gene1	gene2	gene3	gene4
1	0.2	3.9	9.8	1.2
2	3.1	7.6	6.7	2.8
3	5.2	4.1	1.9	13.1
4	1.2	5.1	4.4	11.2
5	1.6	6.3	9.7	14.7

Caso uma das variáveis, após a mutação, tenha valor menor do que zero, o valor deve ser modificado para zero, já que não é possível, nesse caso, existir taxas negativas. Como o objetivo é encontrar valores para X, Y, Z e W que, multiplicados pelos termos totalize 15.7 e a função de avaliação dos indivíduos deve levar em consideração a sua distância em relação à solução, a função de *fitness* para esse problema é descrita por meio da equação 3.23.

$$\mathrm{FI} = \mathrm{Abs}(15.7 - (2 * (\mathrm{gene1}) + 3 * (\mathrm{gene2}) + 4 * (\mathrm{gene3}) - 7 * (\mathrm{gene4})) \quad (3.23)$$

Em que *Abs()* representa o valor absoluto do valor que está entre parênteses.

Dessa forma, o *fitness* de um indivíduo representa a diferença entre o valor desejável (15.7) e o valor que se alcançou com os seus genes e, por isso, quanto falta para se chegar à solução e em quanto ela foi ultrapassada. Portanto, quanto menor o *fitness* do indivíduo mais bem adaptado ele está ao ambiente e mais próximo ele está da solução. Assim, o *fitness* dos indivíduos 1, 2, 3, 4 e 5 são descritos a seguir:

F1=44.

F2=59.7

F3=36.3

F4c=98

F5=148.1.

Para cada um dos 4 indivíduos será criado um novo indivíduo *T*. Esses indivíduos serão chamados aqui de *T1*,*T2*,*T3*, *T4* e *T5*. Para o cálculo da mutação, como se deseja encontrar soluções heterogêneas, será usada a estratégia DE/rand/1 com fator de mutação (*FM*) 0.4. Para o cálculo do vetor *TI* serão usados os indivíduos 2,3 e 5, escolhidos aleatoriamente (hipoteticamente).

Assim,

TI=[3.1 + 0.4*(5.2-1.6) 7.6 + 0.4*(4.1-6.3) 6.7 + 0.4*(1.9-9.6) 2.8+ 0.4*(3.1-14.7)]

TI=[4,54 *6.72* *3,58* *-1.84*]

TI=[4,54 *6.72* *3,58* *0*]

O indivíduo *N1* será gerado a partir do cruzamento entre os indivíduos 1 e *T1*. Nesse exemplo será usado o cruzamento binomial com *Tc* = 0.6. O indivíduo(vetor) 1 pode ser lido como [0.2 3.9 9.8 1.2].

Para a formação do gene 1 do indivíduo *N1*, hipoteticamente, tem-se rand(1) = 0.3. Assim, o gene 1 do indivíduo N1 é igual ao gene 1 do indivíduo *T1*(4.54). Para a formação do gene 2 do indivíduo *N1*, hipoteticamente, tem-se rand(2) = 0.6. Assim, o gene 2 do indivíduo N1 é igual ao gene 2 do indivíduo *T1*(6.72). Para a formação do gene 3 do indivíduo *N1*, hipoteticamente, tem-se rand(3) = 0.8. Assim, o gene 3 do indivíduo N1 é igual ao gene 3 do indivíduo 1(9.8). Para a formação do gene 4 do indivíduo *N1*, hipoteticamente, tem-se rand (4) = 0.1. Assim, o gene 4 do indivíduo N1 é igual ao gene 4 do indivíduo T1(-1.84). Assim:

N1 = [4.54 6.72 9.8 0]

Portanto, o indivíduo *N1* possui *fitness* 39.86. Dessa forma, o indivíduo *N1* tem o *fitness* menor (39.86<44), foi mais bem avaliado e está mais próximo da solução do que o indivíduo 1. Por isso, o indivíduo *N1*sobrevive para a próxima geração e o indivíduo 1 é descartado. Sendo assim, o indivíduo 1 na geração 2 é o indivíduo *N1*.

Após a seleção do primeiro indivíduo para a composição da geração 2, o procedimento descrito nesta seção vai ser repetido para o indivíduo 2, com a formação do indivíduo *N2* e a posterior escolha entre o indivíduo 2 e o *N2* como sobrevivente

para a geração 2. O procedimento também vai ser repetido para os indivíduos 3, 4 e 5. As gerações 3, 4 e 5 são formadas da mesma maneira como foi formada a geração 2.

O objetivo deste trabalho é o desenvolvimento de um método para a criação de classificadores hierárquicos de proteínas. Os classificadores (descritos na seção 3.2) são criados usando a Evolução Diferencial (descrita na seção 3.3). Assim, na seção 3.4, serão descritas as principais características das proteínas, objetos que terão suas funções preditas pelos classificadores.

3.3 Proteínas

Proteínas são cadeias de aminoácidos executoras de atividades essenciais para o organismo de qualquer ser vivo [Alberts et al., 2007] [Devlin, 2011]. Os aminoácidos são formados por um grupo carboxila (COOH), um grupo amina (NH3), uma molécula de Hidrogênio (H) e um radical. Os 20 tipos existentes de aminoácidos se diferenciam pelo radical. O grupo amina é uma base e o grupo carboxila é um ácido. O nome aminoácido vem da junção dos nomes amina e ácido [Nelson and Cox, 2002].

Quando o grupo carboxila de um aminoácido é ligado ao grupo amina de outro, se diz que foi formado um dipeptídeo. Nesse caso, a ligação é chamada de ligação peptídica. Assim, quando há muitos aminoácidos reunidos, se diz que há um polipeptídio. Uma das principais características moleculares das proteínas é o número de cadeias polipeptídicas (ou peptídicas) [Nelson and Cox, 2002].

Se dois aminoácidos são ligados por meio de moléculas de enxofre, se diz que foi formado um dissulfeto. Nesse caso, a ligação é chamada de ligação dissulfeto [Nelson and Cox, 2002].

A síntese de proteínas é resultado de dois processos, chamados de transcrição e tradução. O processo de transcrição usa moléculas de DNA (do inglês *desoxiribonucleic acid*) para sintetizar moléculas de RNA (do inglês *ribonucleic acid*). O processo de tradução usa moléculas de RNA para sintetizar as proteínas [Nelson and Cox, 2002].

As moléculas de DNA são formadas por fitas duplas contendo nucleotídeos, células que contém uma base nitrogenada, uma pentose e um fosfato e podem ser de 4 tipos, que se diferenciam pela base (Adenina (A), Timina (T), Guanina (G), Citosina (C)) (Figura 3.11 (a)). Se, em uma das fitas, o nucleotídeo presente tem base A, na

outra, o nucleotídeo deve ter base T, e vice-versa. O mesmo ocorre para os nucleotídeos de base G e C [Nelson and Cox, 2002].

As moléculas de RNA são sintetizadas a partir do DNA e formadas por fitas simples de nucleotídeos. O RNA, assim como o DNA, possui quatro tipos de nucleotídeos, sendo que ao invés da base T, o RNA usa a base Uracila (Figura 3.11 (b)). Nas moléculas de DNA, se, em uma das fitas, o nucleotídeo presente tem base A, na outra, paralelamente, o nucleotídeo deve ter base U, e vice-versa. O mesmo ocorre para os nucleotídeos de base G e C. Para a formação das moléculas de RNA, a fita é formada tendo como molde uma das fitas do DNA [Nelson and Cox, 2002].

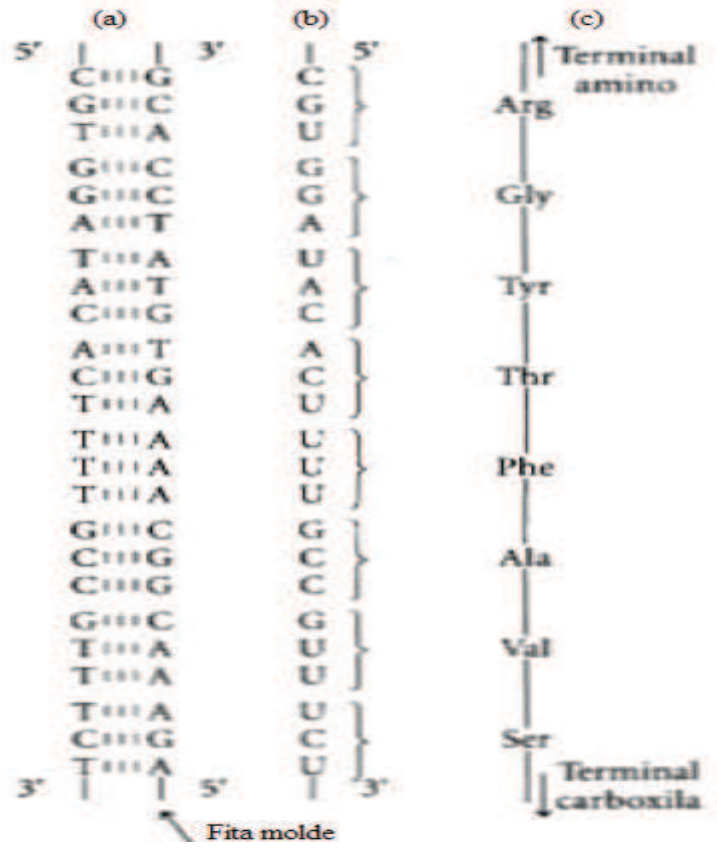

Figura 3.11 Representação das moléculas de DNA(a), de RNA(b) e de proteína [Nelson and Cox, 2002]

Existem três tipos principais de RNA. O RNA mensageiro (mRNA) codifica uma sequência do DNA para síntese de proteínas. O RNA de transferência (tRNA) lê o código presente no mRNA e transfere para uma cadeia usada para a síntese de proteínas e o RNA ribossomal (rRNA) sintetiza as proteínas. Para a síntese de proteínas, cada trecho do RNA é transformado em um códon. Os códons são partes constituintes de um

aminoácido que, por sua vez, são partes constituintes de uma proteína (Figura 3.12 (c)) [Nelson and Cox, 2002].

Por se tratar de moléculas complexas, a estrutura das proteínas foi dividida em quatro níveis de complexidade. Assim, são definidos quatro níveis de estrutura para uma proteína: a estrutura primária, secundária, terciária e quaternária [Freitas e Carvalho, 2007] [Nelson and Cox, 2002].

A estrutura primária representa todas as ligações existentes entre os aminoácidos, dentre elas as ligações peptídicas e dissulfeto. Por meio da representação dessas ligações pode-se conhecer a sequencia dos aminoácidos da proteína, sendo que essa sequência é única para cada proteína (Figura 3.12 (a)) [Nelson and Cox, 2002].

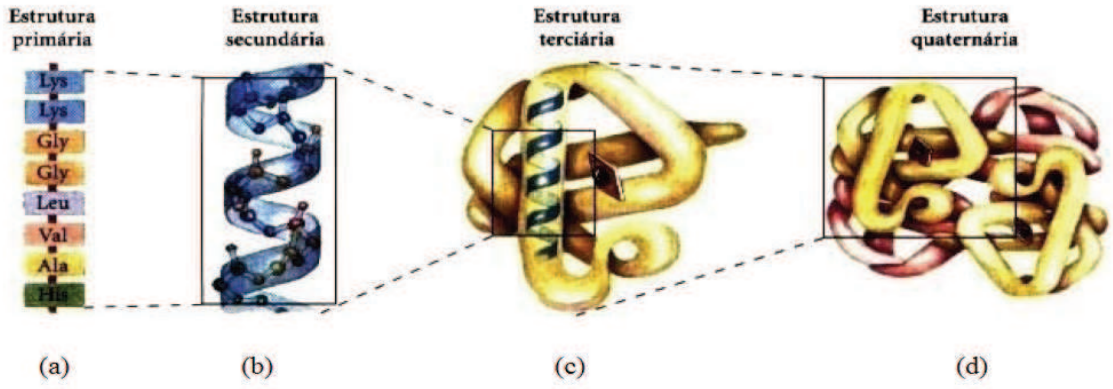

Figura 3.12 Quatro estruturas de uma proteína [Nelson and Cox, 2002]

A estrutura secundária representa os padrões de arranjos de aminoácidos nas proteínas (Figura 3.12 (b)). O polipeptídio resultante pode ter formato helicoidal. Nesse caso, o tipo da estrutura é chamado de α-hélice. Existem outros, como, por exemplo, a conformação β [Nelson and Cox, 2002]. A estrutura terciária, assim como a estrutura secundária, representa os padrões de arranjos de aminoácidos nas proteínas, porém leva em consideração aspectos que envolvem a distância entre os aminoácidos (Figura 3.12 (c)) [Nelson and Cox, 2002]. Caso a proteína tenha mais de uma cadeia polipeptídica, os seus padrões de arranjos de aminoácidos com os aspectos que envolvem a distância entre os aminoácidos serão representados pela estrutura quaternária (Figura 3.12 (d)) [Nelson and Cox, 2002].

Diversas informações acerca das proteínas podem auxiliar na predição das suas funções, como por exemplo os quatro níveis da estrutura citados [Nelson and Cox,

2002] e as características dos genes que deram origem às proteínas [Enright et Al, 1999] [Ettema et al., 2001] [Marcotte, et al., 1999] [Overbeek et al., 1999] [Zheng et al., 2002]. Caso uma proteína em um organismo tenha sido formada a partir da junção de outras duas proteínas em outro organismo, a função das duas proteínas de origem podem ser usadas para a predição da proteína final. Outro fator que auxilia no processo de predição das funções das proteínas é o fato de duas proteínas terem sido sintetizadas por meio de genes vizinhos nos cromossomos de organismo diferentes. [Overbeek et al., 1999] [Ettema et al., 2001] [Zheng et al., 2002].

Dentre os papeis que uma proteína pode assumir estão o de um receptor transmembranar, por meio das proteínas GPCR [Devlin, 2011], catalisador de reações químicas [Alberts et al., 2007] e biológicas, por meio das enzimas, transportador extracelular, por meio da hemoglobina e mioglobulina, protetor, por meio da imunoglobulina, hormonal, por meio da insulina, dentre outros [Devlin, 2011].

Para que sejam classificadas computacionalmente, as proteínas, suas classes e as relações entre as classes precisam ser organizadas e descritas por um sistema que permita o acesso de algoritmos computacionais a elas. Com esse objetivo, foram criadas as ontologias de proteínas, descritas na seção 3.3.1.

3.3.1 Ontologia de Proteínas

Uma ontologia é a especificação de um vocabulário de representação de um domínio compartilhado para discurso. Esse vocabulário engloba atributos de objetos e as classes as quais esses objetos pertencem, assim como as relações entre as classes [Gruber, 1993]. O modelo definido pela ontologia contém também o significado dos atributos, classes e relações [Gruber, 2009].

Algumas das ontologias criadas para classificação da função de proteínas mais usadas são a FunCat (do inglês *Functional Catalogue*) [Ruepp et al., 2004] e a GO (do inglês *Gene Ontology*) [Ashburner et al., 2000], que foram criadas com o objetivo de descrever as proteínas do ponto de vista funcional. Porém, enquanto a FunCat representa a função das proteínas por meio de uma árvore, a GO o faz por meio de um DAG.

A FunCat é definida como um sistema de classificação que permite a descrição das funções de proteínas de qualquer organismo. Cada classe a qual uma proteína

pertence (cada função que a proteína exerce) em um nível da árvore é definida por um número de 2*P dígitos, sendo P o nível do nó ao qual a classe está associada na árvore. No primeiro nível, por exemplo, as funções de metabolizar energia e realizar a transcrição são representadas pelos números 02 e 11, respectivamente. No segundo nível, a classe 1102 representa a função de transcrever moléculas de RNA, enquanto no terceiro nível a classe 110201 representa a função de transcrever moléculas de rRNA [Ruepp et al., 2004]. A classe 110201 pode ser representada pelo nó pintado com a cor cinza na árvore ilustrada na figura 3.13. A FunCat é uma ontologia flexível e escalável. Por isso, a inserção de uma nova função pode ser feita em qualquer nível da árvore sem alterar a consistência do modelo.

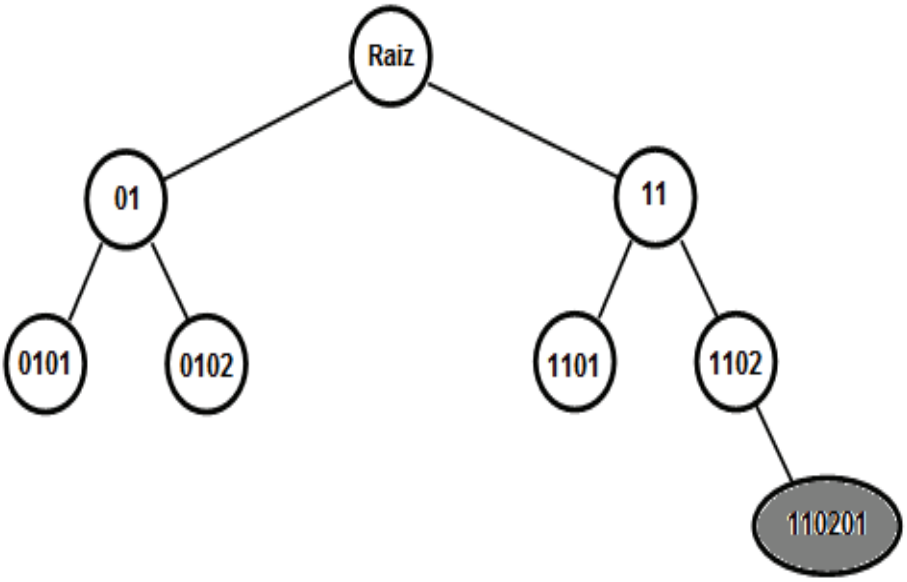

Figura 3.13 Representação de uma classe da ontologia FunCat.

Como os números que representam as funções em cada nível são valores categóricos, eles não são sequenciais e o número da nova função pode ser qualquer número que não esteja sendo usado para outra função [Ruepp et al., 2004].

O objetivo da GO é produzir um vocabulário dinâmico e controlado que possa descrever o papel dos genes e das proteínas. A GO representa as funções das proteínas usando grafos e possui três bancos de dados diferentes que representam três ontologias diferentes. A primeira delas (*Biological Process*) representa os processos biológicos que o DNA e as proteínas contribuem. A segunda (*Molecular Function*) representa as

funções das proteínas e a terceira (*Cellular Component*) a estrutura das células eucariontes onde o DNA e as proteínas agem.

Na GO, como as possíveis funções das proteínas são representadas por um DAG, qualquer nó da estrutura pode ter mais de um pai. Portanto, a GO permite a representação de uma função que pode ser a especificação de outras duas ou mais. As funções são representadas por um identificador, tendo como modelo :GO: + 7 números (figura 3.15).

De acordo com a figura 3.14, o nó associado à classe GO0051444 é filho do nó associado à classe GO0051438 e do nó associado à classe GO0051352 e a classe GO0043086 representa a função de inibição da catálise [Ashburner et al., 2000], que é o processo de aumento da velocidade de uma reação química qualquer.

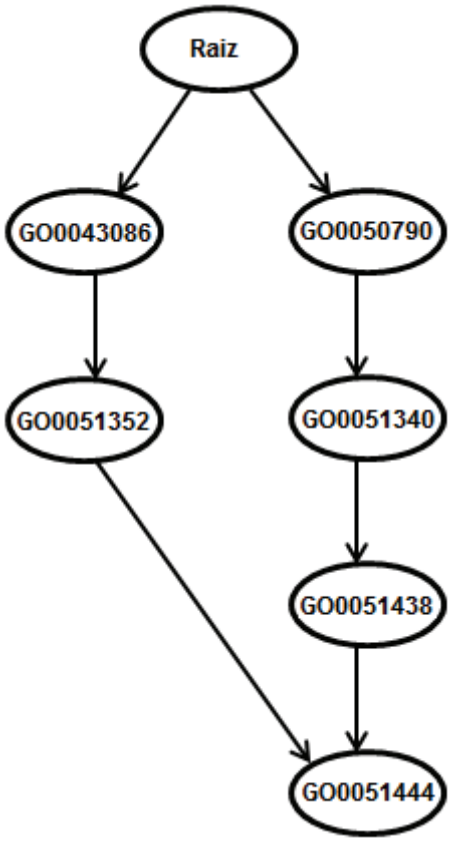

Figura 3.14 Representação da função de uma proteína descrita na ontologia GO.

Na figura 3.14 a classe GO0051352 representa a função de inibição da reação de ligação entre duas substancias quaisquer [Ashburner et al., 2000]. A classe GO0050790 representa a função de estimular a catálise [Ashburner et al., 2000]. A classe GO0051340 representa a função de estímulo da reação de ligação entre duas substancias

quaisquer e a classe GO0051438 representa a função de estimular a ligação entre uma proteína qualquer e a ubiquitina [Ashburner et al., 2000]. A classe GO0051444 representa a função de inibir a ligação entre uma proteína qualquer e a ubiquitina [Ashburner et al., 2000].

3.4 Considerações finais do capítulo

Neste capítulo foram introduzidos os conceitos de classificadores locais e globais, Evolução Diferencial (ED) e proteínas. Esses conceitos são fundamentais para o entendimento da solução proposta (RCM) já que nela um algoritmo baseado na ED é usado para a criação de classificadores globais da função de proteínas.

O processo de desenvolvimento do RCM é detalhado neste trabalho, sendo que outros sete métodos foram criados. Neste capítulo foram detalhados os conceitos de classificadores locais por nó e por nó-pai já que, dos sete métodos criados para o desenvolvimento do RCM, um gera classificadores locais por nó-pai e três geram classificadores locais por nó.

No próximo capítulo são apresentados os principais trabalhos relacionados à ED e à classificação hierárquica de proteínas, principais temas relacionados ao RCM. É importante conhecer os principais métodos relacionados à ED porque ela apresenta limitações e, como a intenção dos autores deste trabalho é o uso da ED com o menor número de limitações possível, serão apresentadas as possíveis soluções para cada problema no capítulo 3 e os argumentos para o uso das soluções escolhidas no capítulo 4.

Como o método aqui proposto deve ser comparado com métodos de mesmo objetivo, alguns dos métodos usados atualmente para classificação hierárquica da função proteica são apresentados no capítulo 3. Nem todos os métodos são usados para comparação nos experimentos (capítulo 5), sendo que a razão para o uso de cada um é descrita no capítulo 5.

Capítulo 4

Estado da arte

Na seção 4.1 são apresentados os principais trabalhos relacionados à ED, técnica de comprovada eficiência na resolução de problemas relacionados a diversas áreas do conhecimento [Coelho, 2008][Panduro, 2009][Salomon et al., 2000] e usada como base para a construção do RCM. Isso faz com que, desde 1996, ano da sua criação, algumas limitações tenham sido encontradas e muitas variações tenham sido criadas [Das et al., 2009][Epitropakis et al, 2010][Zhang and Sanderson, 2009] [Noman, et al., 2011][Liu et al., 2010]. Na seção 4.2 são apresentados as principais abordagens relacionadas à classificação hierárquica de proteínas, tarefa executada pelos classificadores gerados pelo RCM e que tem sido estudada devido a sua importância na resolução de problemas das ciências biológicas [Holden and Freitas, 2008a] [Otero et al., 2009] [Otero et al., 2010].

4.1 Evolução Diferencial

No artigo original da ED [Storn and Price, 1995], duas estratégias foram propostas, conhecidas como DE/rand/1 (equação 3.20) e DE/target-to-best/1 (equação 3.21).Um dos criadores da ED publicou um artigo [Storn, 1996] com duas novas estratégias de uso da técnica, sendo chamadas por ele de: DE/best/1 e DE/best/2. Essas quatro estratégias foram usadas como base para o desenvolvimento de variações da ED com o objetivo de abrandar as deficiências do modelo.

Um dos problemas da ED é que o algoritmo tende a convergir rapidamente para uma solução, que normalmente é o mínimo local [Buhry et al., 2009][Das et al., 2009][Zhang and Sanderson, 2009]. Além disso, o algoritmo pode estagnar em alguma região do espaço de estados, ficando ali indefinidamente ou levando muito tempo para sair [Buhry et al., 2009][Das et al., 2009].

A maioria dos algoritmos de procura baseado em populações tenta alcançar duas metas: velocidade de convergência (*exploitation*) e diversidade da população (*exploration*). As estratégias da ED que usam o melhor individuo da população como base para a mutação (Best/1,Best/2 e rand-to-best/1) tendem a ter maior velocidade de

convergência do que os outros [Das et al., 2009], já que o espaço em torno dos melhores indivíduos é explorado e há maior probabilidade de uma solução estar nesse local. Porém, estudos mostram que o algoritmo pode convergir rapidamente para uma solução que não é a ótima, e sim um mínimo local [Buhry et al., 2009][Das et al., 2009] [Zhang and Sanderson, 2009][Islam et al., 2012].

Quando a estratégia rand-to-best é usada, além do problema citado, caso o resultado da diferença (S1 – S2) seja pequena, o algoritmo tende a não encontrar nenhuma das melhores regiões do espaço de estados, já que os indivíduos gerados serão pouco modificados. A estratégia rand/1, que usa indivíduos aleatórios na mutação, tende a ter a população mais diversificada do que quando as outras são usadas, o que, em alguns casos, faz com que o algoritmo demore mais para encontrar a solução ótima [Epitropakis et al, 2010].

Diante dos problemas citados, algumas variações têm sido criadas para tentar resolvê-los. Em uma delas, propostas por [Das et al., 2009] e denominada DEGL (do inglês *DE with Global and Local Neighborhoods*), os indivíduos são organizados em formato de um anel. Para a mutação de um indivíduo, são criados dois outros indivíduos, sendo um deles baseado em uma mutação local e outro em uma mutação global. Para a formação do indivíduo TI, os dois indivíduos formados (local e global) são usados, junto com uma variável *w*. [Das et al., 2009] propuseram seis formas de se calcular o valor de *w*. O funcionamento do DEGL usando uma das formas de cálculo de *w* (DEGL/SAW) é descrito no capítulo 5.

Nos experimentos do DEGL, o algoritmo foi usado para achar o valor máximo de nove funções. Foram feitos nove experimentos, cada um com uma função diferente e cada um com as seis variações do DEGL. Em todos os experimentos realizados, uma ou mais variações encontrou a solução ótima mais rapidamente do que quando a rand-to-best original foi usado.

Em outra variação, criada por [Buhry et al., 2009], todos os indivíduos da população tem chances de serem usados na mutação. Assim, a probabilidade de um indivíduo ser sorteado para esse fim é inversamente proporcional ao seu *fitness*. Dessa forma, quanto mais apto o indivíduo, mais ele tem chances de ser usado. Portanto, o algoritmo não usa somente o melhor indivíduo, como nas estratégias DE/Best/1, DE/best/2 e DE/rand-to-best. Esse mecanismo faz com que, ao invés de explorar a

área em torno de um indivíduo, o algoritmo explore as partes do espaço de estados onde estão os melhores indivíduos. Ao mesmo tempo o algoritmo não explora áreas de pouco interesse, já que os indivíduos de baixa aptidão têm pouca probabilidade de serem usados. Nos experimentos, a ED foi usada para achar o máximo das funções. Foram feitos experimentos com variações da ED usando as estratégias DE/rand/1, DE/best/1 e uma estratégia chamada no artigo de DE/rand/min, que combina três indivíduos sorteados com o de mais aptidão. Os experimentos foram feitos com cinco funções diferentes, sendo que 2 delas tem mínimo global e 3 tem mínimos local e global. Nos experimentos nos quais só havia mínimos globais, os algoritmos implementados com as estratégias DE/rand/min, DE/best/1 sem a variação tiveram a melhor performance. O autor sugeriu que isso ocorreu porque não houve a possibilidade desses algoritmos ficarem estagnados no mínimo local, já que ele não existe. Porém quando o problema tinha mínimos locais e globais, o algoritmo com a estratégia DE/rand/1 com variação teve o melhor desempenho, atingindo o mínimo global, assim como o DE/rest/1 com variação e o DE/rand/1 original. Os algoritmos usando as estratégias DE/best/1, DE/rand/2 e a variação do DE/rand/2 ficaram estagnados em algum lugar do espaço de estados e não saíram mais.

Outra variação da ED, proposta por [Epitropakis et al, 2010] e denominada ProDE, calcula a distância entre os indivíduos para realizar a mutação. Assim, para a mutação de um indivíduo, a distância entre ele e todos os outros indivíduos é calculada. Após isso, são sorteados três indivíduos, sendo que a probabilidade de cada um ser sorteado é inversamente proporcional a sua distância para o indivíduo a ser modificado. A equação usada é a mesma da estratégia original DE/rand/1, porém os indivíduos não são sorteados aleatoriamente, e sim baseados na distância ao indivíduo a ser modificado. Os autores quiseram usar a velocidade de convergência da estratégia DE/best/1 e a capacidade de explorar todas as áreas do espaço de estado da estratégia DE/rand/1. Segundo os autores, usar somente a estratégia DE/best/1 como base não evitaria a convergência prematura para um mínimo local, já que indivíduos mais próximos ao indivíduo a ser modificado seriam usados. Portanto, é realizado um sorteio no qual os indivíduos mais próximos têm mais chances, mantendo a aleatoriedade que faz com que a estratégia DE/rand/1 percorra todo o espaço de estados. Os experimentos foram feitos com o uso de 6 variações da DE, sendo elas: DE/best/1, DE/rand/1, DE/current-to-best/1, DE/best/2, DE/rand/2, DE/current-t-

best/2. As modificações propostas no trabalho foram aplicadas a essas seis variações, sendo denominadas ProDE/best/1, ProDE/rand/1, ProDE/current-to-best/1, ProDE/best/2, ProDE/rand/2, ProDE/current-t-best/2. Assim, a DE/best/1 foi comparada com a ProDE/best/1, a DE/rand/1 foi comparada com a ProDE/rand/1, e assim por diante. Por meio dos experimentos, os autores concluíram que a ProDE teve desempenho melhor do que a DE do ponto de vista da exploração do espaço de estados. Em relação à velocidade de convergência, a ProDE não obteve benefícios e nem houve deterioração do desempenho em relação à DE.

No algoritmo JADE, proposto por [Zhang and Sanderson, 2009], é proposta uma estratégia de mutação, chamada DE/curent-to-pbest/1, para a qual o melhor indivíduo em um grupo de p% da população deve ser escolhido. O JADE pode ser usado com ou sem arquivo externo. No caso do uso com arquivo, em todas as gerações, todos os indivíduos descartados são armazenados no arquivo, sendo que os indivíduos armazenados neles também são usados para a mutação. Caso o arquivo seja preenchido com um número pre-determinado de indivíduos, alguns indivíduos são aleatoriamente descartados. Em ambos os algoritmos, a cada geração, para cada indivíduo é definida uma taxa de cruzamento. Nos experimentos, o JADE foi comparado com outras cinco versões da ED (sendo uma delas a clássica DE/rand/bin/1) e o algoritmo PSO (do inglês *Particle Swarm Optimisation*). De acordo com os resultados, o JADE se mostrou competitivo.

[Islam et al., 2012] propuseram uma variação do JAD com modificações no cruzamento e nos parâmetros da mutação. Segundo os autores, o objetivo foi criar um algoritmo menos ganancioso (com menor velocidade de convergência) e com maior capacidade de exploração do espaço de estados. No algoritmo proposto por [Islam et al., 2012], o cruzamento binomial é usado, sendo que, para a formação do indivíduo *NI*, ao invés do cruzamento ser feito entre *TI* e *I*, ele é feito entre *TI* e um dos p melhores indivíduos da população, sendo que p decresce na passagem entre as gerações.

Nos experimentos, o algoritmo proposto por [Islam et al., 2012] foi comparado com seis outras versões da ED, que se diferenciam apenas pela estratégia de mutação. Duas dessas versões usam estratégias de mutação clássicas (DE/rand/1/bin e DE/current-to-best/1/bin) e as outras quatro são variações que compõem o estado da

arte ([Qin et al. 2009] [Brest et al., 2006] [Das et al., 2009] [Zhang and Sanderson, 2009]), sendo uma dela o JADE.

Foi proposto por [Brest and Maucec, 2008] uma estratégia para redução do número de indivíduos em uma ED. A redução do número de indivíduos ocorre nas passagens entre algumas gerações. Para a redução, após a avaliação dos indivíduos, les são divididos em dois grupos, sendo o primeiro composto pelos N/2 primeiros indivíduos e o segundo pelos N/2 últimos indivíduos. Assim, o i-ésimo indivíduo de um grupo é comparado com o i-ésimo indivíduo do outro grupo. O indivíduo pior avaliado é descartado. O algoritmo proposto por [Zamuda and Brest, 2012] , assim como a sua extensão, proposta [Zamuda et al., 2013], usaram a estratégia de redução proposta por [Brest and Maucec, 2008]. Na tabela 4.1 são listados os principais trabalho que tentam minimizar as limitações da ED.

Tabela 4.1 Lista dos principais trabalhos que tentam minimizar as limitações da ED.

Referência	**Objetivo**
[Das et al., 2009]	Evitar a convergência precoce das estratégias do tipo best.
[Buhry et al., 2009]	Evitar a convergência precoce das estratégias do tipo best.
[Epitropakis et al., 2010]	Usar a velocidade de convergência das estratégias do tipo best e a exploração do espaço de estado das estratégias do tipo rand.
[Zhang and Sanderson, 2009]	Evitar a convergência precoce das estratégias do tipo best.
[Istam et al., 2012]	Evitar a convergência precoce das estratégias do tipo best.
[Brest and Maucec, 2008]	Diminuir o número de indivíduos da população.

Conforme mencionado nesta seção, as estratégias que usam o indivíduo mais bem avaliado da população como referencia tendem a convergir rapidamente para um máximo local. Na tabela 4.1 são apresentados quatro algoritmos que tem como principal objetivo a minimização do prejuízo causado pela convergência precoce. O uso desse tipo de estratégia é indicado para o método descrito neste trabalho, conforme será detalhado no capítulo 4.

Além da ED, outro assunto diretamente ligado com o tema deste trabalho é a classificação hierárquica de proteínas, já que o RCM constrói classificadores

hierárquicos proteicos. Assim, os principais e mais atuais trabalhos ligados à classificação hierárquica de proteínas são descritos na seção 4.2.

4.2 Classificação Hierárquica

Na tabela 4.2 , 4.3 (métodos globais) e 4.4 (métodos locais) encontram-se os principais trabalhos relacionados à classificação hierárquica de proteínas.

Tabela 4.2 Lista dos principais trabalhos que descrevem métodos que constroem classificadores hierárquicos globais das funções de proteínas e usam árvores para representar a hierarquia entre as classes.

Estrutura		**Nome do método**	**Referência**
Árvore	Monorrótulo	HLCS-Tree	[Romão, 2012]
		h-AntMiner	[Otero et al., 2009]
		GMNB	[Silla and Freitas, 2009]
		GMNBwU	[Silla and Freitas, 2009]
		HC-CNN	[Borges, 2012]
		HC-ES	[Borges, 2012]
	Multirrótulo	MHC-CNN	[Borges, 2012]
		MHC-ES	[Borges, 2012]
		HMC-GA	[Cerri et al., 2012]
		hm-AntMiner	[Otero et al., 2010]
		HLCS-Multi	[Romão, 2012]
		Clus-HMC	[Vens et al., 2008]

Nas tabelas 4.3 e 4.4 são apresentados os principais trabalhos relacionados a classificadores hierárquicos globais para uma hierarquia de classes representada por árvores e DAG´s, respectivamente.

Tabela 4.3 Lista dos principais trabalhos que descrevem métodos que constroem classificadores hierárquicos globais das funções de proteínas e usam DAG´s para representar a hierarquia entre as classes.

DAG	Monorrótulo	HLCS-DAG	[Romão, 2012]
		HC-CNN	[Borges, 2012]
		HC-ES	[Borges, 2012]
	Multirrótulo	MHC-CNN	[Borges, 2012]
		MHC-ES	[Borges, 2012]
		hm-AntMiner	[Otero et al., 2010]
		HLCS-Multi	[Romão, 2012]
		Clus-HMC	[Vens et al., 2008]

Nas tabelas 4.4 são apresentados os principais trabalhos relacionados a classificadores hierárquicos locais, independente da estrutura de classes usada.

Tabela 4.4 Descrição dos principais trabalhos que descrevem métodos que constroem classificadores hierárquicos locais da função de proteínas.

Árvore	Monorrótulo	HEHRS	[Holden and Freitas, 2009]
		LMNBwU	[Silla and Freitas, 2009]
		HLCS-Local	[Romão, 2012]
		Sem denominação	[Secker et al., 2010]
	Multirrótulo	HMC-LMLP	[Cerri and de Carvalho, 2011]
		Clus-HSC	[Vens et al., 2008]
DAG	Multirrótulo	HMC-LMLP	[Cerri and de Carvalho, 2011]
		Clus-HSC	[Vens et al., 2008]

A seguir, todos os trabalhos serão descritos, juntamente com os experimentos usados para as suas avaliações.

Como citado na seção 3.2.1.3, os classificadores por nó-pai têm a desvantagem de propagar um erro cometido para todos os outros níveis abaixo dele. Pesquisadores da Universidade de Kent criaram um método para abrandar essa desvantagem. O método é chamado HEHRS (*Hierarchical Ensemble of Hierarchical Rule Sets*) [Holden and Freitas, 2008a].

Enquanto no classificador local por nó-pai há um classificador (conjunto de regras) para cada nó-pai, no HEHRS há *K* conjuntos de regras para cada nó-pai. Assim, é criado um conjunto para cada nível abaixo dele. Dessa forma, se o classificador estiver sendo criado para um nó do nível um, ou seja, para definir a que classe o exemplo pertence no nível dois, e existirem quatro níveis na estrutura de classes, serão construídos conjuntos de regras para três níveis, sendo eles o nível dois, o nível três e o nível quatro. Nesse caso, os três conjuntos terão regras para predição de classes do segundo nível, mas cada um deles usará exemplos do próprio nível.

A construção das regras no HEHRS é feita com base em uma técnica chamada *bagging*. Nessa técnica, são criados diversos classificadores, cada um usando um algoritmo diferente. Depois disso, os classificadores são combinados diversas vezes, baseado na acurácia das regras. Foi publicada no mesmo artigo duas variações do HEHRS, nas quais as regras são otimizadas usando o algoritmo *PSO* (do inglês *Particle Swarm Optimisation*). Em uma das versões, chamada LimPSO-HEHRS, os pesos das regras possuem limite zero. Na outra versão, denominada PSO-HEHRS, os pesos das regras não possuem limites. Nos experimentos, as três versões do HEHRS foram comparadas com um método baseado no método EMM (do inglês *Extended Multiplicative Method*), chamado em [Holden and Freitas, 2009] de Rule-EMM e com o classificador local por nó-pai original, chamado em [Holden and Freitas, 2009] de *Baseline*. No geral, os algoritmos HEHRS e Rule-EMM apresentaram resultados inferiores ao *Baseline*. Porém, as outras duas versões do HEHRS superaram o classificador local por nó-pai original.

Outro algoritmo usado para a geração de regras de classificação de proteínas foi criado por [Otero et al., 2009]. O algoritmo, denominado hAnt-Miner é baseado no algoritmo ACO (do inglês *Ant Colony Optimisation*). O hAnt-Miner divide a construção de regras em duas colônias, sendo uma delas para a construção dos antecedentes das regras e outra para os conseqüentes. O número de formigas da

colônia de antecedentes é o mesmo da colônia de conseqüentes. Nesse algoritmo, a lista de regras começa vazia e são criadas regras até que um número determinado de exemplos sejam cobertos. Foram feitos experimentos usando cinco bases e as medidas hR, hP e hF. O hAnt-Miner foi comparado com o algoritmo J48 e teve melhor desempenho em 3 das bases.

Inspirado na classificação *top-down* seletiva [Davies et al., 2007], na qual o classificador de cada nó é gerado se usando um algoritmo diferente, pesquisadores da Universidade de Kent [Secker et al., 2010] criaram um algoritmo que gera classificadores nos quais alguns atributos são descartados no momento da classificação. Foram feitos experimentos nos quais o método proposto no trabalho foi comparado com a abordagem *top-down* seletiva. Nos experimentos realizados, o classificador gerado pelo método proposto no artigo realiza predições mais rapidamente do que o gerado pelo algoritmo proposto em [Davies et al., 2007] com desempenho estatisticamente semelhante.

Foram propostos por [Blockeel et, al., 2006] dois algoritmos para a construção de classificadores de funções de proteínas. O primeiro deles é chamado CLUS-SC e gera um classificador plano. O segundo, chamado CLUS-HMC, gera um classificador hierárquico global. Os dois algoritmos geram classificadores que expressam os conhecimentos por meio de árvores de decisão. Os dois métodos são baseados no algoritmo TDIDT (do inglês *Top-Down Induction of Decision Trees*). Usando esse algoritmo, a árvore é construída do nó-raiz e direção aos nós-folha, enquanto a poda é realizada partindo dos nós-folha em direção ao nó-raiz.

Nos experimentos realizados com as classes sendo representadas por árvores (FunCat), o algoritmo Clus-HMC foi comparado com os algoritmos C4.5 e Clus-SC separadamente. Usando a área sob a curva PR, o Clus-HMC superou o C4.5 em todas as 12 bases usadas. Nos experimentos feitos usando os algoritmos Clus-HMC e Clus-SC, o Clus-HMC se mostrou mais eficiente quando o critério analisado foi a velocidade de construção do modelo e classificação dos exemplos da base de testes, interpretação do conhecimento, acurácia [Vens et al., 2008] e área abaixo da curva PR[Blockeel et al., 2006].

Em [Vens et al., 2008], foi proposto o algoritmo Clus-HMC, que gera classificadores hierárquicos locais por-nó de rótulo simples para uma hierarquia de

classes representada por uma árvore ou por um DAG. Além disso, a versão do Clus-HMC para uma hierarquia de classes representada por um DAG foi proposta. Os experimentos foram feitos usando bases de dados das ontologias Funcat e GO. Em todos os experimentos, usando todas as medidas, o Clus-HMC superou o Clus-HSC e o Clus-SC.

[Silla and Freitas, 2009] propuseram três construtores de classificadores hierárquicos baseados no Naive-Bayes. Os classificadores são construídos a partir de bases de dados com a hierarquia entre as classes representada por uma árvore. Dois desses classificadores são globais (GMNG e GMNBwU), sendo que um deles (GMNBwU) considera que erros de classificação em níveis maiores da árvore são mais tolerados do que erros cometidos em níveis menores (*usefulness*). O classificador local (LMNBwU) também leva tolera mais erros em níveis maiores da árvore. Nos experimentos, foram usadas oito bases de dados, sendo quatro delas contendo exemplos de proteínas GPCR e quatro contendo exemplos de enzimas. Foram usadas três medidas, sendo elas hR,hP e hF. Nos experimentos foram analisados os fatores localidade (GMNBwU x LMNBwU) e *usefulness* (GMNB x GMNBwU). Nos experimentos realizados, o GMNBwU foi estatisticamente superior ao LMBNwU e ao GMNB. Apesar disso, no geral, o GMNBwU teve maior revocação hierárquica e o GMNB maior precisão hierárquica.

Foram desenvolvidos por [Romão, 2012] quatro algoritmos para construção de classificadores hierárquicos. Os algoritmos, baseados nos Sistemas Classificadores (LCS, do inglês Learning Classifier Systems), foram denominados HLCS-Local, HLCS-Tree, HLCS-DAG e HLCS-Multi. O HLCS-Local constrói classificadores locais por nó de rótulo simples para uma hierarquia de classes representada por uma árvore. O HCLS-Tree constrói classificadores globais de rótulo simples para uma hierarquia de classes representada por uma árvore. O HLCS-DAG constrói classificadores globais de rótulo simples para uma hierarquia de classes representada por um DAG. O HLCS-Multi tem duas versões. A primeira delas constrói classificadores globais multirrótulo para uma hierarquia representada por uma árvore e a segunda constrói classificadores globais multirrótulo para uma hierarquia representada por um DAG.

Nos experimentos, o HLCS-Local foi comparado com o algoritmo *RIPPER*. Os experimentos foram executados a partir de três bases de dados e com o uso de três medidas, totalizando nove experimentos. Em seis experimentos o *HLCS-Local* superou o ou teve desempenho semelhante ao RIPPER.

O HLCS-Tree foi comparado com o GMNB [Silla and Freitas, 2009]. Os experimentos foram executados a partir de 8 bases, divididas em dois grupos. O primeiro grupo é composto por exemplos de proteínas GPCR e o segundo de enzimas. Segundo o autor de [Romão, 2012], o HLCS-Tree teve desempenho melhor nas bases com exemplos de proteínas GPCR. O HLCS foi comparado com o hAnt-Miner a partir de quatro bases, todas contendo exemplos de canais iônicos (proteínas). Apesar de o HLCS ter superado o hAnt-Miner em alguns experimentos, não houve diferença estatisticamente significativa entre os resultados. A primeira versão do HLCS-Multi (Árvore) foi comparada HLCS-Tree e o superou. A segunda versão foi comparada com o HLCS-DAG e também o superou. As duas versões foram comparadas ao Clus-HMC. As duas versões do HLCS-Multi foram superadas pelo Clus-HMC.

Foram propostos por [Borges, 2012] dois algoritmos para construção de classificadores hierárquicos globais para uma hierarquia de classes representada por um DAG. Os algoritmos, chamado MHC-CNN e MHC-ES, são baseados em redes neurais. Os dois algoritmos se diferenciam pela etapa de aprendizagem da rede. Enquanto o MHC-CNN usa a aprendizagem competitiva o MHC-ES usa uma estratégia evolucionária para aprendizagem da rede. Durante o desenvolvimento do MHC-CNN e do MHC-ES foram criadas as versões HC-CNN e HC-ES, para problemas de rótulo simples.

Nos experimentos, usando bases da FunCat, o HC-CNN superou o HC-ES. O Clus-HSC superou os algoritmos HC-CNN e HC-ES na maioria dos experimentos. Ambos os algoritmos foram comparados ao Clus-HSC Com o uso de 10 bases de dados da GO, o MHC-CNN foi comparado ao MHC-ES, ao Clus-HMC e ao Clus-HSC. O MHC-ES também foi comparado ao Clus-HMC e ao Clus-HSC. O algoritmo MHC-CNN teve resultados significativamente melhores do que o MHC-ES na maioria dos experimentos. Usando a medida AUPRC, os algoritmos Clus-HMC e Clus-HSC tiveram resultados superiores aos algoritmos MHC-CNN e MHC-ES. Usando a medida de revocação, os algoritmos MHC-CNN e MHC-Es tiveram

resultados superiores aos algoritmos Clus-HMC e Clus-HSC. O algoritmo HC-CNN foi comparado com os algoritmos GMNB e GMNBwu em [Borges et al., 2013]. Nos experimentos, a diferença entre os três algoritmos foi estatisticamente insignificante. Apesar disso, o GMNBwu superou estatisticamente o HC-CNN.

O algoritmo HMC-LMNP (*Hierarchical Multi-label Classification with Local Multi-Layer Perceptron*) constrói para cada nível da estrutura de representação da hierarquia entre as classes uma rede neural diferente. A rede neural é do tipo *perceptron* de múltiplas camadas e o treinamento de cada rede é feito usando o algoritmo *back-propagation*. NO HMC-LMLP, cada rede possui três camadas, sendo elas a de entrada, a escondida (*hidden*) e a de saída. A camada de saída da rede do nível *n* da estrutura de classes é a camada de entrada do nível *n*+1. Cada neurônio da camada de saída é associado a uma classe diferente.

Durante a fase de testes, em cada rede, após a entrada dos dados da instância a ser classificada, caso o valor encontrado para uma determinada saída seja maior do que um limiar determinado, a classe associada àquela saída é atribuída à instância. Após a classificação de um novo exemplo o algoritmo elimina as inconsistências nas quais uma classe é atribuída à instância no nível n e a sua classe pai não é atribuída ao exemplo no nível n-1.

Nos experimentos, o HMC-LMLP foi comparado aos algoritmos Clus-SC, Clus-HSC e Clus-HMC. Usando a medida AUPRC, o HMC-LMLP superou os algoritmos Clus-SC e Clus-HSC em 80% dos experimentos e foi superado pelo Clus-HMC em 100% dos experimentos.

O HMC-AG, baseado nos Algoritmos Genéticos, constrói classificadores hierárquicos globais para uma hierarquia de classes representada por uma árvore. O classificador é composto por regras. Sendo assim, cada indivíduo representa uma regra. No HMC-DAG, o indivíduo é formado por *NANT* antecedentes, sendo *NANT* o número de atributos presentes na base de dados. Cada antecedente é formado por quatro genes, sendo eles o operador, dois limiares (um inferior e um superior) e um *flag*, que indica se aquele antecedente precisa ser testado (1) ou não (0). Para atributos categóricos o operador pode ser = (igual) ou ≠ (diferente). Para atributos contínuos os operadores podem ser >(maior), < (menor), ≥ (maior ou igual) ou ≤ (menor ou igual).

O consequente de cada regra é formado por um vetor contendo a probabilidade de o exemplo pertencer a cada uma das classes.

4.3 Considerações finais do capítulo

Neste capítulo foram apresentadas as principais limitações da ED, assim como as suas possíveis soluções. Os seis algoritmos apresentados na tabela 4.1 tentam resolver três problemas diferentes apresentados pela ED. Os algoritmos apresentados em [Das et al., 2009], [Zhang and Sandersom, 2009] e [Islam et al., 2012] tentam resolver o problema da rápida convergência do algoritmo para um máximo local. Os algoritmos se diferenciam na forma como a mutação é feita e consequentemente nos indivíduos escolhidos para participar da mutação. Os algoritmos propostos em [Zhang and Sandersom, 2009] e [Islam et al., 2012] são similares entre si, sendo que a mutação feita nos dois é similar à realizada quando a estratégia DE/target-to-best é usada. No trabalho proposto por [Das et al., 2009], a mutação é feita com base em dois indivíduos , sendo que cada um desses indivíduos usa quatro outros indivíduos como base na sua formação.

No capítulo 3 também são apresentadas as principais abordagens para a classificação hierárquica de proteínas, sendo que muitas delas se diferenciam na forma de exploração da estrutura de classes (local ou global), no número de funções que uma mesma proteína pode executar (monorrótulo ou multirrótulo) e na estrutura de classes usada para representar a hierarquia entre as classes (árvore ou DAG). Algumas dessas abordagens serão comparados ao RCM de acordo com os três fatores citados.

No capítulo 4 os componentes do RCM e o funcionamento de cada um deles são descritos.

Capítulo 5

Descrição da proposta

O RCM é um método baseado na ED para construção de um classificador global multirrótulo para uma hierarquia de classes representada por um DAG. Como parte do processo de desenvolvimento do RCM, outros sete métodos foram criados. Cada método é identificado por três letras após o nome RCM e o símbolo "-". A primeira letra é definida pelo tipo de abordagem usada em relação à forma como a estrutura de classes é explorada para a classificação de um novo exemplo, podendo ser *L*(do inglês *local*) ou *G* (do inglês *Global*). A segunda letra informa a quantidade de classes as quais cada exemplo pode pertencer, podendo ser *S* (do inglês *Single*) ou *M* (do inglês *Multi*). A terceira letra ilustra o tipo de estrutura usada para representar a relação entre as classes, podendo ser *T* (do inglês *Tree*) ou *D* (do acrônimo em inglês *DAG*). Dessa forma, o método RCM-LST é usado para construção de um classificador local de rótulo simples usando a ED para uma hierarquia de classes representada por uma árvore.

O RCM é o primeiro método que usa a Evolução Diferencial para a construção de classificadores hierárquicos. A ED foi escolhida por ser uma técnica de comprovada eficiência em problemas de otimização com valores contínuos, sendo agora em um problema com valores discretos. A representação do conhecimento será feita por meio de regras do tipo SE condições ENTÃO conclusão devido ao fato de elas serem expressivas e de fácil compreensão. Essas duas características são desejáveis para o RCM pois os classificadores gerados serão usados por biólogos na tomada de decisão durante o processo de classificação, já que o entendimento do conhecimento usado pelo classificador facilitaria o processo de tomada de decisão.

Os métodos que criam classificadores de rótulo simples (RCM-LST, o RCM-GST, o RCM-LSD e o RCM-GSD) são descritos nas seções 5.1, 5.2, 5.3 e 5.4, respectivamente, enquanto os métodos que constroem classificadores multirrótulo (RCM-LMT, o RCM-GMT e o RCM-LMD) são descritos nas seções 5.5, 5.6, 5.7, respectivamente. O RCM, principal método deste trabalho, é descrito na seção 5.8.

5.1 RCM-LST

O objetivo do RCM-LST é a extração de regras de classificação de proteínas a partir de uma base de dados com exemplos já rotulados. As regras extraídas formam um classificador hierárquico local de rótulo simples para uma hierarquia de classes representada por uma árvore. O RCM-LST usa a abordagem local por nó-pai. Sendo assim é construído um classificador para cada nó não folha.

A descrição feita a partir daqui leva em consideração a construção do classificador associado a um nó da árvore de classes, sendo a construção dos outros feita de forma idêntica. O conjunto de regras é chamado aqui de *Cr* e o algoritmo que gera as regras chamado de *Gr*. *Cr* é iniciado sem nenhuma regra e preenchido com uma regra por vez (*r*) por *Gr*. São componentes do RCM-LST as variáveis *Ecr* (número de exemplos cobertos por *r*), *Neb* (número de exemplos presentes na base de dados), *Nrn* (número de regras geradas consecutivamente com *Ecr* = 0) e *Max* (número máximo de *Nrn* para critério de parada do método) (Figura 5.1).

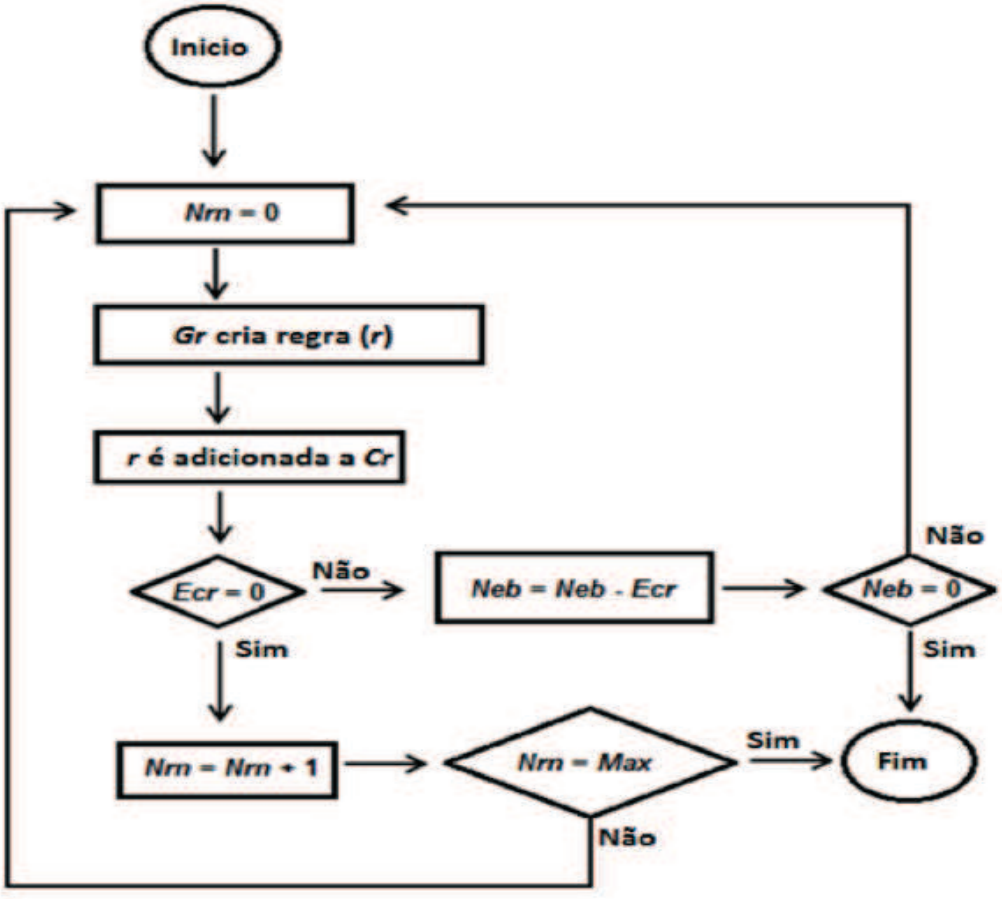

Figura 5.1 Fluxograma do método de geração de classificadores.

Quando *r* é adicionada a Cr, todos os exemplos cobertos por ela são excluídos da base de dados. Por isso, *Neb* é subtraído de *Ecr* (Figura 5.1). Sendo assim, uma regra é criada levando-se em consideração apenas os exemplos não cobertos pelas regras já presentes no classificador. Caso *r* não tenha coberto nenhum exemplo na base de dados (*Ecr* = 0), *Nrn* é acrescida de uma unidade. Caso contrário, como *Nrn* armazena o número de regras geradas consecutivamente com *Ecr*=0, *Nrn* é zerada (Figura 5.1).

De acordo com a figura 5.1, existem dois critérios de parada para o método. Um deles é o número de exemplos presentes na base de dados. Assim, quando não houver mais exemplos na base a geração de regras é finalizada. O outro é o número regras geradas consecutivamente sem cobrir nenhum exemplo da base. Portanto, o método também é finalizado quando *Nrn* alcançar *Max*. O *Gr*, usado para a criação de cada uma das regras, é um algoritmo baseado na Evolução Diferencial e descrito na próxima seção.

5.1.1 O gerador de regras (*Gr*)

O Gr é composto por uma população de *N* indivíduos. Acerca dessa população, se pode definir os parâmetros *min* (número mínimo de genes por indivíduo), *max* (número máximo de genes por indivíduo) e *G* (número de gerações).

[Das et al., 2009] sugere que o número de indivíduos da população seja 10 vezes o número de genes de um indivíduo. Como neste método o número de genes varia entre *max* e *min*, o valor de *N* é calculado da seguinte forma:

$$\boldsymbol{N = \left(\frac{max+\min}{2}\right) * 10} \qquad (5.1)$$

G é um parâmetro livre e depende do problema que está sendo resolvido. O valor de G deve variar de acordo com o valor de *N* [Brest e Maucec, 2008]. Cada indivíduo no *Gr* é representado por uma regra do tipo SE condições ENTÃO conclusão. Para cada indivíduo *I*, têm-se as variáveis $numgenes_I$ e $genes_I$, que tem seus valores escolhidos aleatoriamente, dentro do domínio do problema, além das variáveis $genesinativos_I$, fit_I. *numgenesI* ($min <= numgenes_I <= max$) representa o número de genes (antecedentes) de um indivíduo (regra) e $genes_I$ o conjunto desses genes. $genes_I \in CAtr_I$, sendo $CAtr_I$ o conjunto dos atributos dos exemplos presentes na base de dados. $genesinativos_I = \{CAtr_I - genes_I\}$. Para que o método não tenha prejuízo

na sua velocidade de execução, fazendo com que uma das características da ED seja perdida, apenas os valores dos elementos do conjunto $genes_I$ sofrem mutação. Os elementos do conjunto $genesinativos_I$ são usados apenas para o cálculo da mutação e cruzamento de outros indivíduos, como descrito na seção 1.1.2.1.1.Assim, os *numgenesI*-1 primeiros genes de um indivíduo representam os antecedentes da regra, enquanto o último (*numgenes*-ésimo) representa o consequente.

Fit_I é a função usada para avaliação do indivíduo *I*. No RCM-LST, é calculada como descrito na seção 1.1.3. O *Gr* tem a mesma estrutura e procedimentos da ED original. Os indivíduos (regras) da primeira geração são formados, dentro do domínio do problema, aleatoriamente. Esse processo é descrito na seção 5.1.1.1. As regras, independentemente da geração na qual estejam, são avaliadas como descrito na seção 5.1.1.2. Da segunda geração em diante os indivíduos são gerados a partir de um processo de seleção idêntico ao da ED original. O processo de formação do novo indivíduo (N_I) é descrito na seção 5.1.1.3.

5.1.1.1 Formação da primeira geração

Na primeira geração os genes de cada indivíduo são gerados com base em um dos exemplos da base de dados, para que seja garantido que a regra vai cobrir pelo menos 1 exemplo. O exemplo a ser levado em consideração deve ser sorteado para que haja diversificação na criação dos indivíduos da primeira geração. Caso os valores da base sejam discretos, os genes do indivíduo vão assumir os valores dos atributos do exemplo sorteado. Caso os atributos presentes na base sejam contínuos, eles devem ser discretizados antes da execução do *Gr*.

Com a discretização, para cada atributo serão geradas faixas de valores, sendo que o valor de um atributo de qualquer instância na base de dados deve estar dentro de uma das faixas geradas para aquele atributo. Nesse caso, cada condição no *Gr* será composta por um limite inferior e um limite superior, sendo que um valor satisfaz à condição se ele for maior ou igual ao limite inferior e menor ou igual ao limite superior. Como na ED a evolução dos indivíduos é feita por meio da subtração entre os genes dos indivíduos, cada condição no *Gr* é representada por um número inteiro. O número é atribuído a cada faixa de maneira sequencial. Portanto, a faixa que engloba os menores valores são representadas pelo número 0 (zero), a faixa que engloba os próximos valores é representada pelo número 1, e assim por diante. Para a

formação dos indivíduos da primeira geração, um exemplo da base é sorteado e cada gene do indivíduo vai assumir o número que representa a faixa que engloba o valor do atributo correspondente do exemplo. Assim, caso o exemplo 7 seja sorteado, o gene 1 do indivíduo 1 vai assumir o número que representa a faixa que engloba o valor do atributo 1 do exemplo 7, o gene 2 do indivíduo vai assumir o número que representa a faixa que engloba o valor do atributo 2 do exemplo 7. Para exemplificar o procedimento de formação dos indivíduos da primeira geração, será usada a base 1, descrita na tabela 5.1 e composta por valores discretos:

Tabela 5.1 Descrição dos exemplos para exemplificação da formação dos indivíduos da primeira geração do RCM-LST.

Exemplo/Atributo	A	B	C	Classe
1	1	1	1	1.1
2	2	3	3	2.1
3	1	2	3	1.1
4	1	0	0	2.1
5	1	0	3	2.2

Portanto, o antecedente da regra representada pelo indivíduo 1 é representado pela seguinte expressão: Se Atributo A = 1. O consequente da regra será a classe de um dos exemplos cobertos por ela na base de dados. Caso a regra não cubra nenhum exemplo, a sua classe deve ser escolhida aleatoriamente, respeitando a estrutura de classes.

O indivíduo 1 cobre os exemplos 1,3, 4 e 5 da base 1, já que o valor do atributo A desses exemplos é igual ao único antecedente da regra representada pelo indivíduo1. A classe do indivíduo 1 será 1.1 já que, hipoteticamente, a classe do exemplo 3 foi sorteada. Portanto, o indivíduo 1 criado representa a regra que pode ser definida da seguinte forma: Se atributo A=1 então Classe = 1.1

5.1.1.2 Avaliação das regras

Quando o problema usa classes com relação de hierarquia entre elas, a comparação entre duas classes pode ser feita considerando duas sub-árvores, como descrito na equação 3.7. Esse tipo de comparação é feita porque as duas classes

podem ser completamente iguais, completamente diferentes, mas também podem ter elementos em comum. Caso a comparação fosse feita sem levar em consideração os ancestrais das classes, só haveria as possibilidades de elas serem completamente iguais ou completamente diferentes.

Como descrito na seção 3.2.1.1.3, todos os exemplos usados para a construção de um classificador local por nó pertencem ao mesmo nível da árvore de classes e possuem o mesmo nó-pai. Como as classes de cada uma das regras são escolhidas com base nos exemplos da base de treinamento, elas pertencem ao mesmo nível dos exemplos na árvore de classes e possuem o mesmo nó-pai. Portanto, quando a classe de qualquer regra for comparada à classe de qualquer exemplo, ou elas serão idênticas ou vão se diferenciar apenas no último nível. Considerando que, nesse caso, só existem duas situações, a hierarquia existente entre as classes foi desconsiderada.

Assim, como pode ser visto por meio da equação 5.2, quando a classe prevista for igual à classe de um exemplo que ela cobriu, a regra é avaliada positivamente. Caso contrário, ela é avaliada negativamente. Caso somente os acertos fossem levados em conta, as regras de menor especificidade e consequente maior cobertura tenderiam a ter maior *fitness*, já que elas cobrem mais exemplos e por isso tem mais chances de acertar. Porém, por cobrir muitos exemplos, pode ser que a regra também erre muito. Ter regras com essas características é prejudicial ao classificador, já que, quando ela for usada para classificação, a tendência é que erre muito também. Portanto, as regras são avaliadas de acordo com a equação 5.2:

$$\boldsymbol{Fit_I = NCP/(1 + NCN)} \qquad (5.2)$$

Em que NCNC é o número de casos em que as classes prevista é igual à classe correta e NSNC é o número de casos em que a classe prevista é diferente da classe correta.

O indivíduo 1 é usado para exemplificar a forma de avaliação do método e pode ser lido da seguinte forma: Se Atributo A = 1 e Atributo C = 3 então Classe = 1.1 Para que seja exemplificado o uso da regra, será usada a base de testes descrita na tabela 4.1 e o indivíduo gerado na seção anterior (5.1.1.1), chamado aqui de indivíduo 1. Portanto, o indivíduo 1 cobre os exemplos 1, 3, 4 e 5. Para o cálculo de F1, a classe prevista pelo indivíduo será comparada às classes dos exemplos 1,3,4 e 5.

NCP =1, já que a classe cp (1.1) ea classe do exemplo 1 (1.1) são iguais.

NCP = 2, já que a classe cp (1.1) e a classe do exemplo 3 (1.1) são iguais.

NCN = 1, já que a classe cp (1.1) e a classe do exemplo 4 (1.2) são diferentes.

NCP = NCP + 1, já que a classe cp (1.1) e a classe do exemplo 5 (1.1) são iguais.

Portanto,

Fit_1= 3/(1+1).

Fit_I = 3/2

Fit_1=1.5

Após a avaliação o dos indivíduos de uma geração, novos indivíduos são formados para a formação da geração seguinte. Esse processo é descrito na seção 5.1.1.3.

5.1.1.3 Formação do novo indivíduo (*NI*)

A formação do novo indivíduo (*NI*) é feita em duas etapas. A primeira delas é a formação dos antecedentes da regra representada por ele. A segunda etapa é a formação do consequente da regra. Os antecedentes da regra representada pelo indivíduo *NI* são formados a partir da mutação e do cruzamento. A mutação é feita como proposto por [Das et al., 2009] e o tipo de cruzamento usado é o exponencial. O procedimento de formação dos antecedentes das regras representadas pelo indivíduo *NI* é descrita na seção 5.1.1.3.1.

A mutação do consequente da regra é feita por meio da atribuição de uma classe, já que não pode ser feita como na ED original, a partir de operações matemáticas. O impedimento ocorre porque aqui o consequente é representado por uma sub-árvore (dependendo da versão do método, um sub-grafo).

Para a formação dos antecedentes da regra que representa o indivíduo N_I, inicialmente é realizada uma mutação no indivíduo I e formado o indivíduo T_I. O valor de $numgenes_{TI}$ são sorteados, assim como os elementos do conjunto $genes_{TI}$. Os valores dos genes que representam os atributos presentes em $genes_{TI}$ são calculados por meio da mutação. Os valores dos genes que representam os atributos pertencentes a $genesinativos_{TI}$ serão iguais aos valores dos genes que representam os atributos correspondentes no indivíduo I. O processo de mutação dos genes que representam os

atributos pertences a $genes_{TI}$, assim como a atribuição dos genes que representam os atributos pertencentes a $genesinativos_T$ são descritos na seção 5.1.1.3.1.1. Como o indivíduo T_I é usado apenas para a formação dos antecedentes de N_I, $min<=numgenes_{TI}<=max$-1.

A mutação dos genes que representam as condições da regra é feita como na ED. Como o objetivo do algoritmo a cada vez que é executado é encontrar a regra mais bem avaliada, as estratégias DE/best/1 e DE/best/2, propostas por [Storn, 1996] ou a estratégia DE/current-to-best proposta por [Storn and Price, 2005] poderiam ser usadas aqui. O uso delas seria adequado porque como o objetivo aqui é encontrar apenas 1 regra (indivíduo), superar o *fitness* do indivíduo mais bem avaliado é o objetivo de todos os outros indivíduos da população. Uma forma de se fazer isso é tentando adquirir características semelhantes às desse indivíduo, como feito nas 3 estratégias citadas. Porém, quando qualquer uma das três estratégias de mutação citadas é usada, o algoritmo pode convergir rapidamente para o máximo local [Islam et al., 2012][Das et al., 2009]. Com o objetivo de resolver esse problema foi proposto por [Das et al., 2009] um algoritmo chamado DEGL/SAW(do *inglês DE with Global and Local Neighborhoods/Self Adaptive Weight*). No DEGL/SAW, cada indivíduo possui um peso, chamado *w*, em que seu valor é modificado a cada geração. A estratégia de mutação usada neste trabalho é a mesma usada no DEGL/SAW. Nela, os indivíduos são organizados no formato de um anel, conforme ilustrado na figura 5.2. No anel, os indivíduos são organizados na ordem de criação da primeira geração.

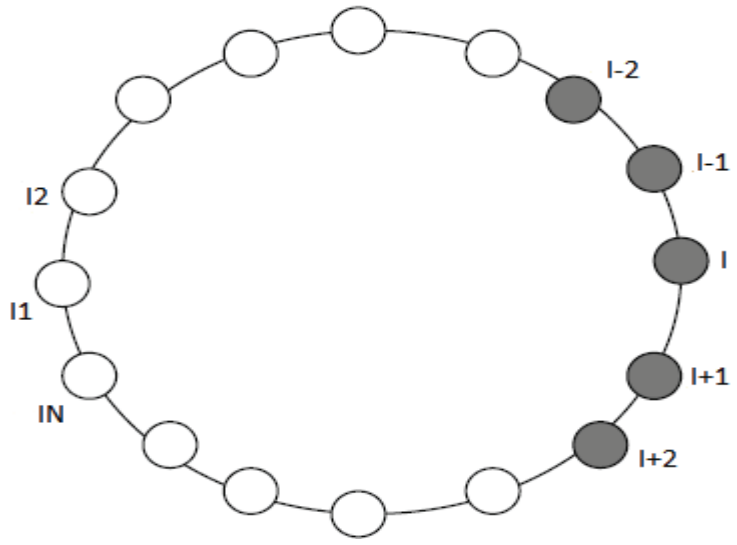

Figura 5.2 Ilustração da organização dos indivíduos no algoritmo proposto por [Das et al., 2009]. Adaptado de [Das et al., 2009].

Na figura 5.2 estão marcados de cinza o indivíduo *I* e os seus vizinhos com raio de vizinhança (*rv*) 2. No DEGL/SAW, para o cálculo da mutação do individuo *I* e a consequente formação do indivíduo T_I são criados um indivíduo baseado em uma mutação local, chamado aqui de LT_I, um indivíduo baseado em uma mutação global, chamado aqui de GT_I e um fator de mutação do indivíduo T_I, chamado aqui de w_{TI}.

No RCM-LST, o fator de mutação de um indivíduo o acompanha na passagem das gerações, assim como acontece no DEGL/SAW. Para o cálculo de GT_I são sorteados dois indivíduos quaisquer em toda a população, chamados aqui de R_1 e R_2. O indivíduo GT_I é formado de acordo com a equação 5.3.

$$\boldsymbol{GT_I = I + \alpha * (M - I) + \beta * (R_1 - R_2)} \quad (5.3)$$

Em que *M* é o indivíduo mais bem avaliado da população e β e α são dois fatores de multiplicação.

Para o cálculo de LT_I são sorteados dois vizinhos em um determinado raio de vizinhança (*rv*). Os dois vizinhos locais são chamados aqui de *p* e *q*. O indivíduo LT_I é formado de acordo com a equação 5.4

$$\boldsymbol{LT_I = I + \alpha * (MV - I) + \beta * (p - q)} \quad (5.4)$$

Em que *MV* é o vizinho do indivíduo *I* mais bem avaliado.

Os autores do DEGL sugerem que o valor de *w* para os indivíduos da primeira geração seja escolhido aleatoriamente entre 0 e 1. Para os indivíduos TI, esse valor é calculado de acordo com a equação 5.5:

$$\boldsymbol{w_{TI} = w_I + FM * (w_B - w_I) + F * (w_{R1} - w_{R2})} \quad (5.5)$$

Em que w_B é o valor de *w* do indivíduo de melhor *fitness* da população, w_{R1} o *w* de R_1 e w_{R2} o *w* de R_2. O valor máximo de w_{TI} é 0.95 e o valor mínimo 0.05.

O cálculo do vetor T_I é feito como descrito na equação 4.6:

$$\boldsymbol{T_I = w_{TI} * G_{TI} + (1 - w_{TI}) * L_{TI}} \quad (5.6)$$

Para cada posição dos vetores GT_I, LT_I e T_I:

Se $R >= X + 0.5$ então $R = X+1$

Se $R < X + 0.5$ então $R = X$

Se $R>U$ então $R = R – U$

Se $R< 1$ então $R = (R* (-1))$

Em que R é o valor presente em cada elemento do vetor, e U é valor máximo que ao elemento pode assumir.

A classe da regra representada pelo indivíduo T_1 é escolhida da mesma forma que as classes das regras que os indivíduos da primeira geração representam são escolhidas. Para exemplificar o uso do método será usada uma população composta pelos seguintes indivíduos (valores hipotéticos):

Indivíduo 1:

$numgenes_1 = 2$, $genes_1$ = {Atributo A, Atributo C}, $genesinativos_1$= {Atributo B}, Fit_1 (*fitness* do indivíduo 1) = 10 e $w_1 = 0.1$. Assim, os antecedentes da regra representada pelo indivíduo 1 pode ser definida como: Se Atributo A=1 e Atributo C = 0. $W_1 = 0.1$. Os valores dos atributos presentes em *genesinativos1* são: Atributo B = 1

Indivíduo 2:

$numgenes_2 = 2$, $genes_2$ = {Atributo B, Atributo C}, $genesinativos_2$ = {Atributo A}, $Fit_2 = 1$ e $w_2 = 0.05$. Assim, os antecedentes da regra representada pelo indivíduo 2 pode ser definida como: Se Atributo B = 2 e Atributo C = 3. Os valores dos atributos presentes em $genesinativos_2$ são: Atributo A = 2

Indivíduo 3:

$numgenes_3 = 1$, $genes_3$ = {Atributo C}, $genesinativos_3$ = {Atributo A, Atributo B}, $Fit_3 = 12$ e $w_3 = 0.44$. Assim, os antecedentes da regra representada pelo indivíduo 3 pode ser definida como: Se Atributo C = 3. Os valores dos atributos presentes em $genesinativos_3$ são: Atributo A = 3 e Atributo B = 3

Indivíduo 4:

$numgenes_4 = 3$, $genes_4$ = {Atributo A, Atributo B, Atributo C}, $genesinativos_4$ = {}, $Fit_4 = 11$, $w_4 = 0.13$. Assim, os antecedentes da regra representada pelo indivíduo 4 pode ser definida como: Se Atributo A=1 e Atributo B = 2 e Atributo C = 1.

Indivíduo 5:

$numgenes_5$ = 3, $genes_5$ = {Atributo A, Atributo B, Atributo C}, $genesinativos_5$= {}, Fit_5 = 22, w_5 = 0.76. Assim, os antecedentes da regra representada pelo indivíduo 5 pode ser definida como: Se Atributo A=2 e Atributo B = 2 e Atributo C = 3.

Indivíduo 6:

$numgenes_6$=1, $genes_6$ = {Atributo A}, $genesinativos_6$ = {Atributo B, Atributo C}, fit_6 = 20, w_6 = 0.9. Assim, os antecedentes da regra representada pelo indivíduo 6 pode ser definida como: Se Atributo A=1. Os valores dos atributos presentes em $genesinativos_6$ são: Atributo B = 0 e Atributo C = 1

Indivíduo 7:

$numgenes_7$ = 1, $genes_7$ = {Atributo B}, $genesinativos_7$ = {AtributoA, Atributo C}, Fit_7 = 13 e w_7 = 0.34. Assim, os antecedentes da regra representada pelo indivíduo 7 pode ser definida como: Se Atributo B = 1. Os valores dos atributos presentes em $genesinativos_7$ são: Atributo A = 3 e Atributo C = 2

Indivíduo 8:

$numgenes_8$ = 2, $genes_8$ = {Atributo A, Atributo C}, $genesinativos_8$={Atributo B}, Fit_8 = 2, w_8 = 0.1. Assim, os antecedentes da regra representada pelo indivíduo 8 pode ser definida como: Se Atributo A=2 e Atributo C = 0. Os valores dos atributos presentes em $genesinativos_8$ são: Atributo B = 1.

Os indivíduos do exemplo foram organizados de acordo com a ilustração da figura 5.3. Foi sugerido por [Das et al., 2009] que *rv* seja 10% do valor de *N*. Como nesse exemplo *N*=8, então *rv*=0.8. Como não existe 0.8 indivíduos, *rv* foi aproximado para 1. [Das et al., 2009] sugere que os valores de *F*, α e β sejam iguais. Nesse exemplo, *F*= α=β=0.2. Para os cálculos a seguir, o Atributo A é representado pelo gene 1, o Atributo B é representado pelo gene 2 e o Atributo C é representado pelo gene 3. Inicialmente, *I*=1 e, por isso, o individuo a sofrer mutação é o indivíduo 1. Portanto, será criado um novo indivíduo (T_1) para ser comparado a ele. Por meio de sorteio tem-se os seguintes valores: $numgenes_{TI}$ = 2, $genes_T$ = {Atributo A, Atributo B} e $genesinativos_T$ = {Atributo C}. Assim, somente os Atributos A e B vão sofrer mutação. Como *rv* = 1, cada indivíduo tem dois vizinhos. No caso do indivíduo 1, os seus vizinhos são os indivíduos 2 e 8. Portanto, *p* = indivíduo 2 e *q* = indivíduo 8.

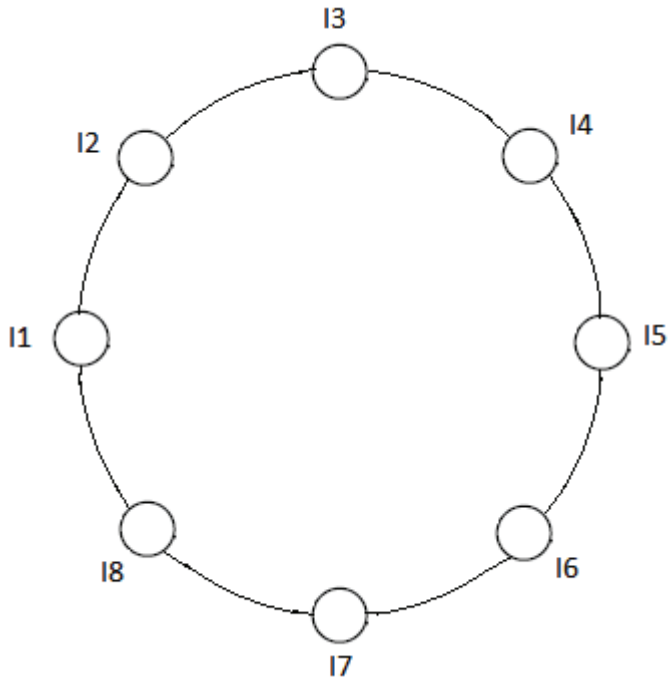

Figura 5.3 Ilustração da organização dos indivíduos em um exemplo do algoritmo proposto.

Os indivíduos usados para a formação do indivíduo GT_1 são R_1, R_2, o indivíduo que está sofrendo mutação (indivíduo 1) e o indivíduo de melhor *fitness* da população (indivíduo 5). Hipoteticamente, os indivíduos R_1 e R_2 foram sorteados como sendo, respectivamente, os indivíduos 3 e 6. Para o cálculo do indivíduo GT_1, tem-se:

$GT_1=[I1 + \alpha * (I5 – I1) + \beta * (I3 – I61)]$

Considerando que dois genes sofrerão mutação:

$G_{T1}=[I1_1+\alpha*(I5_1–I1_1)+\beta*(I3_1 – I6_1) \quad I1_2+\alpha*(I5_2–I1_2)+\beta*(I3_2 – I6_2)]$

Em que I_g é o gene g do individuo I.

$GT_1=[1 + 0.2*(2 – 1)+0.2*(3 –1) \qquad 1 + 0.2*(2 – 1)+0.2*(3 –0)]$

$GT_1 = [1.6 \qquad 1.8]$

$GT_1=[2 \qquad 2]$

Portanto, a regra que representa o indivíduo GTI é: Se Atributo A=2 e Atributo B = 2. Para a criação de *LT* e a consequente mutação do indivíduo 1, são usados os indivíduos *p*,*q*, o vizinho mais bem avaliado do indivíduo 1 (indivíduo 8) e o indivíduo que está sofrendo mutação (indivíduo 1). Os indivíduo p e q foram

sorteados hipoteticamente como sendo os indivíduos 2 e 8. Para o cálculo do indivíduo LT1, tem-se:

LT_1= [I1+α*(I81–I1)+ β*(I2 – I8)]

LT_1= [$I1_1$+α*($I8_1$–$a1_1$)+ β*($I2_1$ – $I8_1$) $a1_2$+α*($I8_2$–$I1_2$)+ β*($I2_2$ – $I8_2$)]

LT_1 =[1 +0.2*(2 – 1)+0.2*(2 –2) 1 +0.2*(1 – 1)+0.2*(2 –1)]

LT_1 = [1.2 1.2]

Usando o caso 2 da tabela 1.1,

LT_1 =[1 1]

Portanto, a regra representada pelo indivíduo LT é: Se Atributo A=1 e Atributo B = 1. Para o cálculo de w_{T1} são usados os valores de *w* dos indivíduos 5 (indivíduo mais bem avaliado da população) , 3 e 6 (indivíduos R_1 e R_2, respectivamente):

$w_T = w_1+F*(w_5-w_1)+F*(w_3-w_6)$

$w_T = 0.1+0.2*(0.76-0.1)+0.2*(0.44-0.9)$

$w_T = 0.14$

Para o cálculo do valor do gene 1 de *T* são usados w_{T1} e os indivíduos GT_1 e LT_1.

T_1= [$_{wT}$*$GT1_1$+(1- wT)*$LT1_1$ wT*$GT1_2$+(1-wT)*$LT1_2$]

T_1 = [0.14*2+(1-0.14)*1 0.14*2+(1-0.14)*1]

T_1 = [1.14 1.14]

Usando o caso 2 da tabela 1.1, T_1 = [1 1]

Como o Atributo C pertence ao conjunto $genesinativos_{T1}$, o seu valor no indivíduo T_1 é o mesmo valor do Atributo C do indivíduo 1. Portanto, Atributo C = 0. Por meio da mutação do indivíduo 1, o indivíduo T_1 passa a ser composto pelas condições correspondentes aos valores discretos 1,1 e 0. Portanto, a regra representada pelo indivíduo T_1 tem os seguintes antecedentes: Se (Atributo A= 1) e (Atributo B=1) e (Atributo C=0).

Como somente os Atributos *A* e *B* pertencem ao conjunto T_T, o Atributo C não é levado em consideração no uso da regra com o objetivo de se classificar um objeto. Portanto, desconsiderando o atributo C, os antecedentes da regra representada pelo indivíduoT_I é definida como: Se (Atributo A =1) e (Atributo B=0).

Após o indivíduo T_I ser gerado por meio da mutação do indivíduo *I*, N_I será gerado a partir do cruzamento entre os indivíduos *T* e *I*. Só vai ocorrer cruzamento entre os Atributos que pertençam a T_T.

Os valores dos conjuntos $numgenes_{NI}$, $genes_{NI}$ e $genesinativos_NI$ são os mesmos dos conjuntos $numgenes_{TI}$, $genes_T$ e $genesinativos_T$, respectivamente. O valor de w_{NI} é o mesmo de w_{TI}.

O exemplo usado nesta seção é uma continuação do exemplo usado na seção anterior. Portanto, o indivíduo *NI* será o resultado do cruzamento dos indivíduos 1 e *T*. Para o indivíduo 1, tem-se os seguintes valores: $numgenes_I$=2, $genes_I$={Atributo A, Atributo C } e $genesinativos_I$={Atributo B}. Assim, os antecedentes da regra que o Indivíduo 1 representa podem ser definidos da seguinte forma: Se Atributo A = 1 e Atributo C = 0. O Atributo B tem valor 1.

Como $numgenes_{NI}$=$numgenes_{TI}$, $genes_{NI}$=$genes_{TI}$ e $genesinativos_{NI}$=$genesinativos_{TI}$ e $numgenes_{TI}$=2,$genes_{TI}$={Atributo A, Atributo B} e $genesinativos_{TI}$={Atributo C}, então $numgenes_{NI}$=2, $genes_{NI}$={Atributo A, Atributo B} e $genesinativos_{NI}$={Atributo C}. Assim, somente os Atributos A e B vão participar do cruzamento, que é feito como descrito na seção 2.3.3.2. Como o *fitness* do Indivíduo 1 é 10 e esse valor representa 50% de 20 (valor do *fitness* máximo que um indivíduo pode atingir-valor hipotético), *Tc* = 0.5. Para o cruzamento entre os indivíduos 1 e T1 foram sorteados os valores rand (1) = 0.7 e rand (2) = 0.3. Como 0.7*(rand (1))>0.5*(*Tc*), o primeiro gene do indivíduo *NI* terá o mesmo alelo do gene correspondente do indivíduo 1. Como 0.3*(rand (1))<=0.5*(*Tc*), o segundo gene do indivíduo *NI* terá o mesmo alelo do gene correspondente do indivíduo *T*. O terceiro gene não sofrerá cruzamento, já que pertence ao conjunto $genesinativos_{NI}$. Assim, ele vai ter o mesmo valor do terceiro gene do indivíduo 1.

Para o indivíduo N_I, $numgenes_{NI}$=2, T_I={Atributo A, Atributo B} e $geneinativos_I$ = {Atributo C}.Portanto, o indivíduo N_I será representado pela seguinte regra: Se Atributo A = 1 e Atributo B = 3. Como $w_{NI} = w_{TI}$, w_{NI} = 0.14. O Atributo C

tem valor 0. Como foi realizada a formação dos antecedentes da regra que representa o indivíduo N_I, o próximo passo é a formação do consequente. A classe que compõe o consequente do indivíduo N_I é a classe de um dos exemplos que o indivíduo cobre. O indivíduo N_I cobre os exemplos 3 e 5, que pertencem às classes 1.1 e 2.2, respectivamente. Hipoteticamente, a classe o exemplo N_I foi escolhida como sendo a classe 1.1 (classe do exemplo 3).

Nessa seção foi descrito o RCM-LST, algoritmo local de rótulo simples com a estrutura entre as classes representada por uma árvore. Na seção 5.2 o RCM-GST é descrito, que se diferencial no RCM-LST apenas na forma como a estrutura de classes é percorrida para se realizar a classificação, que no caso do RCM-GST é de maneira global.

5.2 RCM-GST

O RCM-GST constrói classificadores hierárquicos globais de rótulo simples da função de proteínas para uma hierarquia de classes representada por uma árvore. O classificador é formado por regras extraídas a partir de uma base de dados com exemplos já rotulados. Assim, o método constrói um conjunto de regras para toda a hierarquia de classes. A construção do classificador com o uso do RCM-GST é feita da mesma maneira como feito na construção do classificador de um nó no uso do RCM-LST. Porém, para a mutação do consequente do indivíduo N_I após se escolher a classe de uma das regras que o indivíduo cobre é feito um procedimento para modificação dessa classe. Esse procedimento é descrito na seção 5.2.1. A avaliação das regras no RCM-GST é feita como descrito na seção 5.2.2.

5.2.1 Modificação da classe do indivíduo *NI*

A modificação da nova classe do indivíduo N_I é feita com base em seis parâmetros. Os seis parâmetros, descritos na seção 5.2.1.1. A forma como a mutação é feita é descrita na seção 5.2.2.2.

5.2.1.1 Cálculo dos 6 parâmetros

Os seis parâmetros descritos nesta seção são usados como base para a substituição e foram criados pelos autores deste trabalho. Para a realização do cálculo são usados todos os exemplos da base cobertos pelo indivíduo. Assim, a classe

prevista pela regra é comparada à classe de cada exemplo coberto. Ao se comparar duas classes hierárquicas, existem sete situações possíveis. As classes podem não ter nenhum ancestral em comum (completamente diferentes), podem ser idênticas, e podem ser parcialmente diferentes. Essa última situação se divide em cinco novas situações. O número de vezes que cada uma dessas situações acontece, com exceção da primeira, é um parâmetro usado para a atribuição da nova classe do indivíduo. Um dos seis parâmetros usados para a mutação das classes é a contagem de exemplos cobertos pela regra quando a classe prevista é idêntica à classe correta (conhecida). A referência a esse parâmetro será feita por meio das letras AT (Acerto Total) (Figura 5.4).

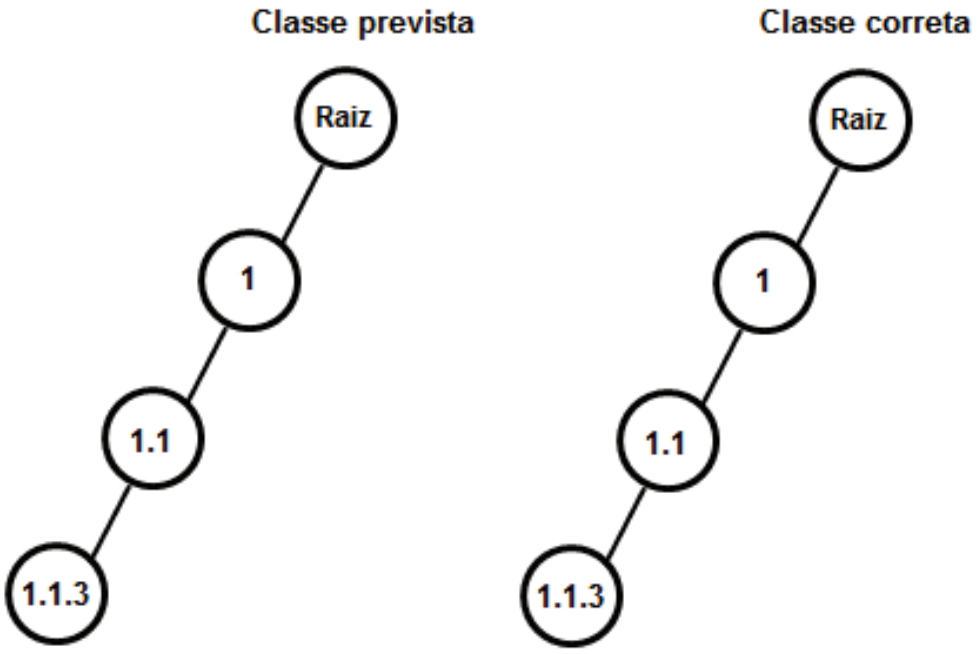

Figura 5.4 Ilustração de um acerto para o parâmetro T.

Nas figuras 5.4 a 5.17 terão ilustrações de trechos de árvores para que o entendimento dos procedimentos seja facilitado. Assim, a classe prevista será representada apenas pelos nós que representam a classe prevista e os seus ancestrais, com os outros nós da estrutura de classes descartados da figura. O mesmo será usado para a classe correta.

No exemplo da figura 5.4, a classe prevista (1.1.3) é idêntica à classe correta (1.1.3). O segundo parâmetro a ser utilizado é a contagem de exemplos cobertos pela regra quando a classe prevista é igual a algum ancestral da classe correta (conhecida), que será referenciado nesse trabalho como ACF (Acerto Com Falta) (Figura 5.5).

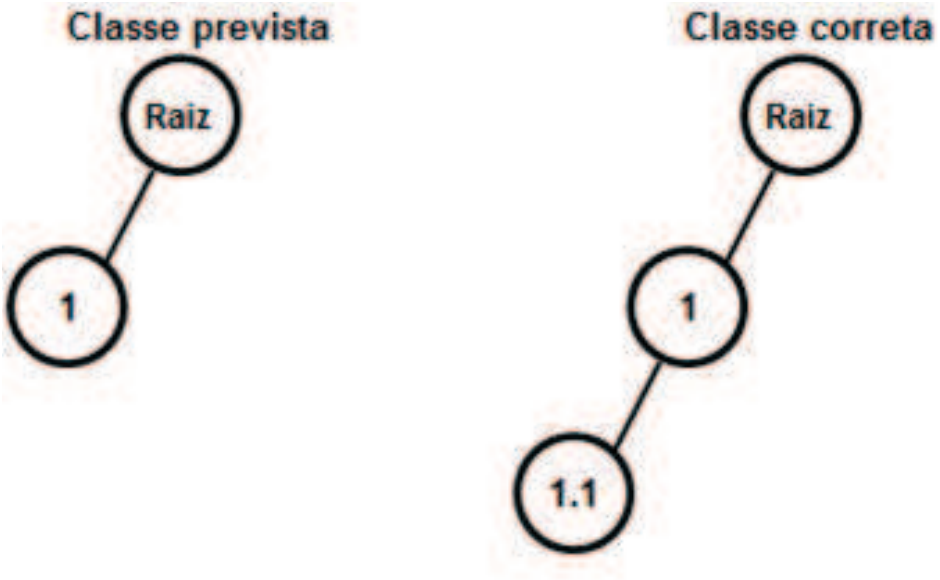

Figura 5.5 Ilustração de um acerto para o parâmetro ACF.

Como pode ser observado no exemplo da figura 5.5, a classe prevista é igual ao ancestral 1 da classe correta (1.1).

O terceiro parâmetro a ser utilizado é a contagem de exemplos cobertos pela regra quando as classes prevista e correta são diferentes, têm ancestrais em comum e as duas tem o mesmo número de níveis. Esse parâmetro será referenciado como AP (Acerto Parcial) (Figura 5.6).

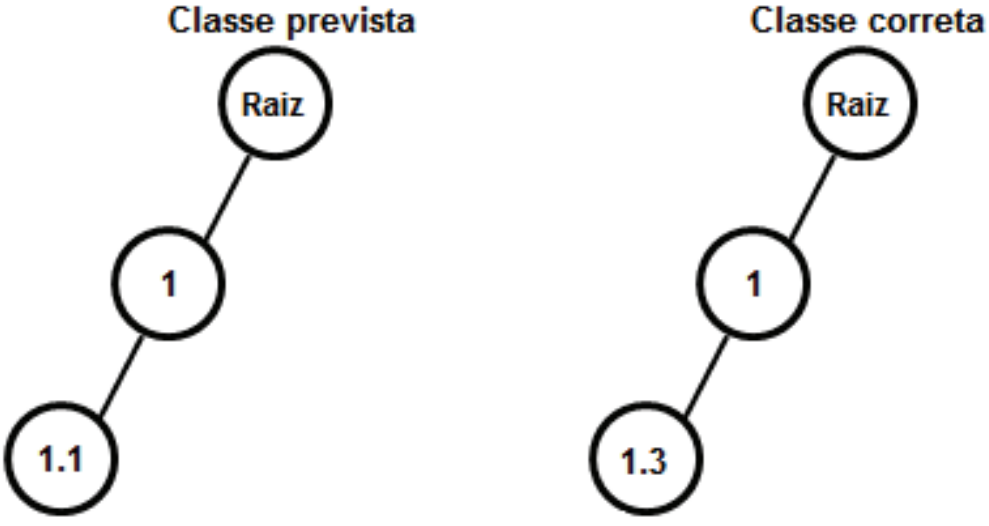

Figura 5.6 Ilustração de um acerto para o parâmetro AP.

No exemplo da figura 5.6 a classe prevista tem o ancestral 1 em comum com a classe correta (1.3) e as duas classes tem o mesmo número de níveis (3).

O quarto parâmetro usado é a contagem de acertos quando a classe correta (conhecida) é igual a um ancestral da classe prevista e será referenciado como ACE (Acerto Com Excesso) (Figura 5.7).

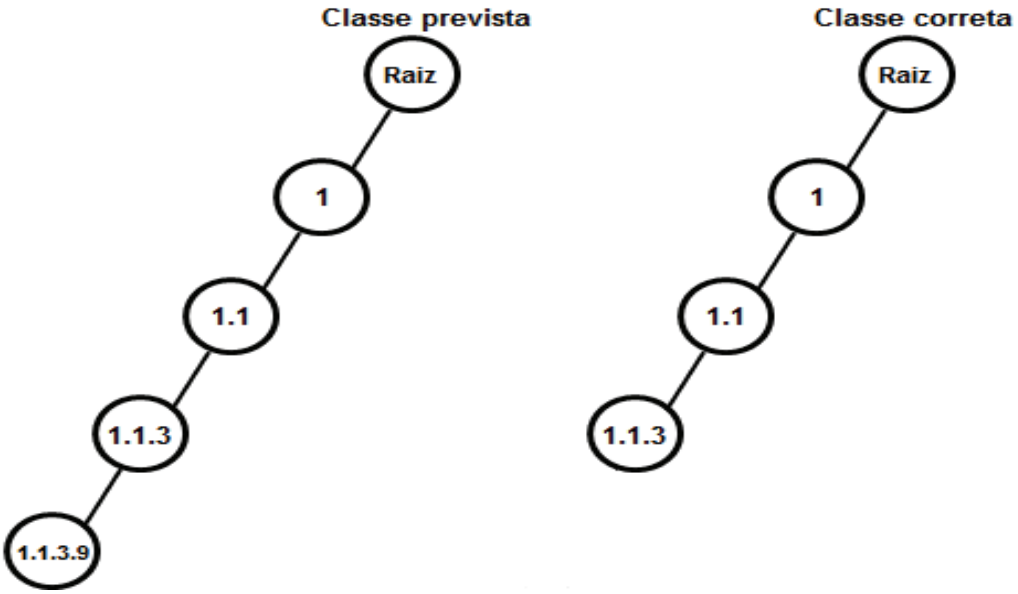

Figura 5.7 Ilustração de um acerto para o parâmetro ACE.

No exemplo da figura 5.7, a classe correta é igual ao ancestral 1.1.3 da classe prevista (1.1.3.9). O quinto parâmetro a ser utilizado é mostrado na figura 5.8.

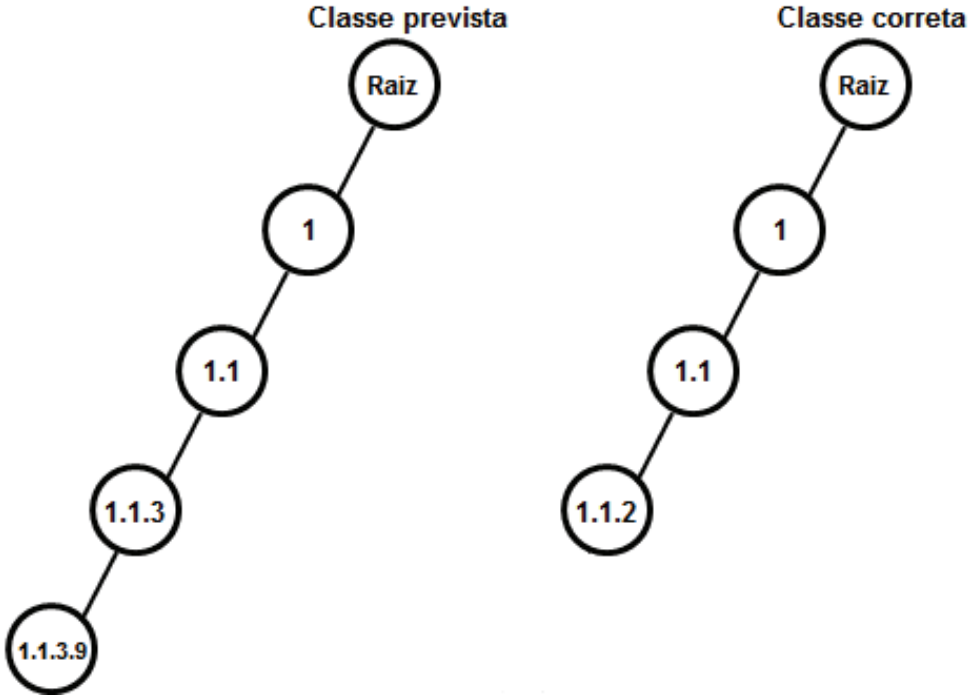

Figura 5.8 Ilustração de um acerto para o parâmetro ACEE.

Esse parâmetro é a contagem de exemplos cobertos pela regra quando a classe prevista tem mais níveis do que a classe correta, a classe correta não é igual a nenhum ancestral da classe prevista, mas elas têm ao menos um ancestral em comum. Esse parâmetro será referenciado como ACEE (Acerto Com Excesso e Erro) (Figura 5.8). No exemplo da figura 5.8 a classe prevista tem mais níveis do que a classe correta (cinco e quatro, respectivamente), a classe prevista (1.1.3.9) não tem nenhum ancestral igual à classe correta (1.1.2), mas as duas possuem os ancestrais 1 e 1.1 em comum.

Ô sexto parâmetro a ser utilizado é chamado aqui de ACFE (Acerto Com Falta e Erro). Um exemplo da ocorrência de um ACFE é ilustrado na figura 5.9. O parâmetro ACFE é a contagem de exemplos cobertos pela regra quando a classe correta tem mais níveis do que a classe prevista, a classe prevista não é igual a nenhum ancestral da classe correta, mas elas têm ao menos um ancestral em comum. (Figura 5.9).

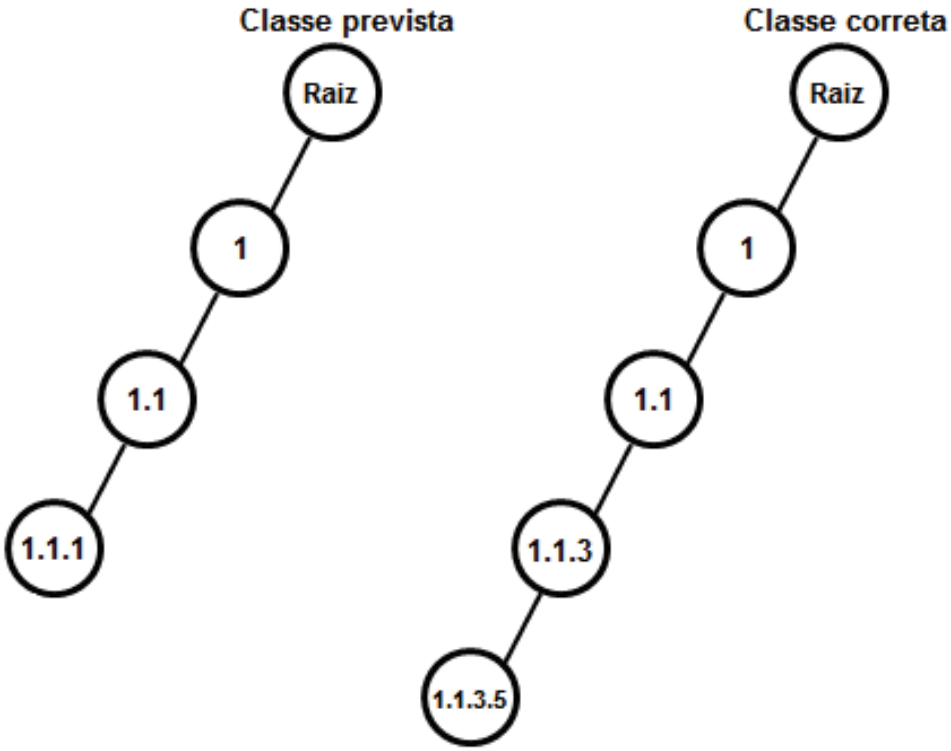

Figura 5.9 Ilustração de um acerto para o parâmetro ACFE.

No exemplo da figura 5.9 a classe correta tem mais níveis do que a classe prevista (5 e 4, respectivamente), a classe prevista (1.1.1) não é igual a nenhum ancestral da classe correta (1.1.3.5), mas elas têm os ancestrais 1 e 1.1 em comum.

Com os 6 parâmetros descritos, na seção 4.2.2.2 é descrita a forma como as classes são modificadas, baseadas nos parâmetros descritos nesta seção.

4.2.2.2 Mutação das classes

Após o término da contagem dos parâmetros, a classe de cada regra será modificada. As ações a serem tomadas a partir do cálculo dos parâmetros têm como objetivo maximizar o parâmetro AT do indivíduo.

Caso os parâmetros AT ou ACE tenham tido o maior valor entre todos os parâmetros (vencedor), a classe do indivíduo não precisa ser modificada. No primeiro caso na maior parte dos exemplos a classe prevista já está igual à correta. No segundo, a classe prevista não precisa ser modificada, pois a classe correta é um antecedente da classe prevista e, por isso, até onde é conhecida a classificação a classe foi prevista de maneira correta.

Caso o vencedor seja o ACF, para que na próxima geração a regra tenha maior probabilidade de ter mais acertos, a sua classe precisa ter *n* ancestrais a mais. O valor de *n* é calculado como descrito na equação 5.7:

$$n = (\sum_{i=1}^{X} B_i)/X \tag{5.7}$$

Em que *X* o número de casos onde houve ACF.

$$B_i = ncci - ncp \tag{5.8}$$

Em que *ncci* o número de níveis da classe correta do exemplo *i* e *ncp* o número de níveis da classe prevista.

Se o parâmetro AP for o vencedor, para que, na próxima geração, a probabilidade de o novo indivíduo ter mais acertos aumente, a classe precisa ter seus *x* últimos níveis modificados. O valor de *x* é calculado como descrito na equação 5.9:

$$x = (\sum_{i=1}^{Y} C_i)/Y \tag{5.9}$$

Em que *Y* é o número de casos onde houve AP .

$$C_i = ncci - naci \tag{5.10}$$

Em que *ncci* o número de níveis da classe correta do exemplo i e *naci* o número de ancestrais comuns entre as classes prevista e correta do exemplo *i*.

Caso o parâmetro ACEE seja o vencedor, a classe prevista deve ter menos *c* níveis e, dos remanescentes, os *c* últimos serão modificados. O valor de c é calculado como descrito na equação 4.11 e o valor de m como descrito na equação 5.13.

$$c = (\sum_{i=1}^{W} H_i)/W \tag{5.11}$$

Em que W o número de casos onde houve ACEE.

$$H_i = ncp - ncci \tag{5.12}$$

$$m = (\sum_{i=1}^{Z} Ki) \tag{5.13}$$

$$K_i = ncci - naci \tag{5.14}$$

Caso o parâmetro ACFE seja o vencedor, a classe prevista deve ter seus últimos *f* níveis modificados e, depois, mais *d* níveis.

$$f = (\sum_{i=1}^{U} Q_i)/U \tag{5.15}$$

Em que U o número de casos onde houve ACFE.

$$Q_i = ncp - naci \tag{5.16}$$

$$d = (\sum_{i=1}^{R} L_i)/R \tag{5.17}$$

$$L_i = ncci - ncp \tag{5.18}$$

Após a definição da ação a ser tomada e do número de nós a serem modificados ou acrescentados, os novos nós são escolhidos por meio de sorteio, respeitando as restrições da estrutura da árvore de classes. Para exemplificar o método proposto, considera-se a classe de quatro regras. A classe da primeira delas é ilustrada na figura 5.10, com as seguintes porcentagens dos parâmetros:

AT=10%

AP=15%

ACF=40%

ACE = 10%

ACEE=15%

ACFE=10%

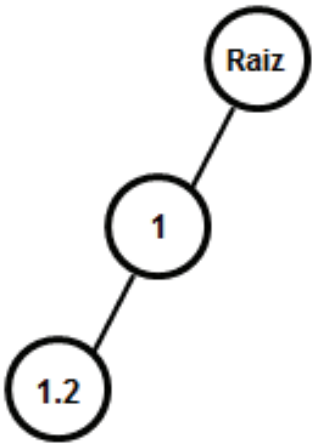

Figura 5.10 Classe da primeira regra antes da modificação.

Dessa forma, o parâmetro vencedor foi o ACF. Como *n*=1 (valor hipotético), a classe desse indivíduo vai ter 1 nível a mais. Assim, um nó-filho do nó 1.2 foi sorteado e a classe da regra ficou como mostrado na figura 5.11.

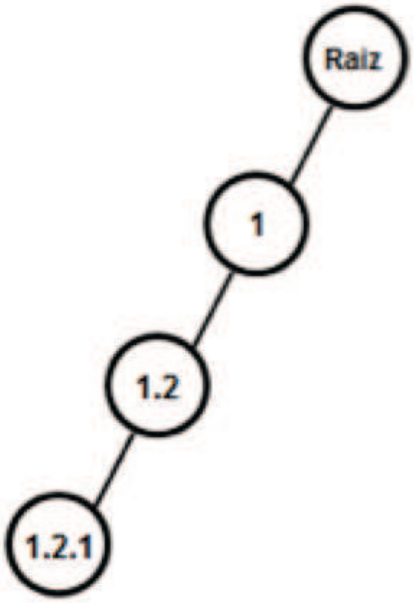

Figura 5.11 Classe da primeira regra depois da modificação.

A classe do indivíduo passou a ter 1 nível a mais. Antes era 1.2 (2 níveis) e passou a ser 1.2.1 (3 níveis). A classe da segunda regra usada para demonstração do método é mostrada na figura 5.12. A regra 2 teve as seguintes porcentagens dos parâmetros:

AT=5%

AP=35%

ACF=20%

ACE = 5%

ACEE=20%

ACFE=15%

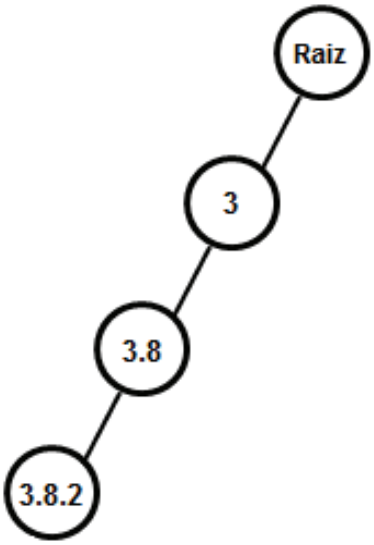

Figura 5.12 Classe da terceira regra antes da modificação.

Dessa forma, o parâmetro ganhador foi o AP, com a classe após a modificação ilustrada na figura 5.13.

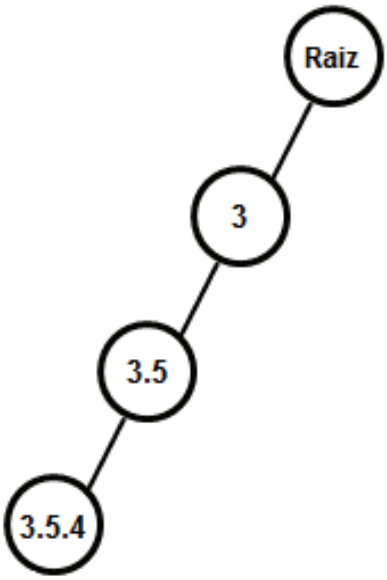

Figura 5.13 Classe da terceira regra depois da modificação.

Como $x = 2$ (valor hipotético), os nós dos últimos dois níveis (3.8 e 3.8.2) da classe desse indivíduo serão modificados. Após o sorteio, a classe da regra 2 ficou como mostrado na figura 5.13. O nível 2 estava associado à classe 3.8, sendo modificada para 3.5. A partir daí uma classe associada a um nó filho de 3.5 foi sorteada, sendo ela 3.5.4. Portanto, a classe da regra 2 foi modificada parcialmente, já que era 3.8.2 e passou a ser 3.5.4. A classe da terceira regra usada para demonstração do método é mostrada na figura 5.14. A regra 2 teve as seguintes porcentagens dos parâmetros:

AT=5%

AP=20%

ACF=10%

ACE = 20%

ACEE=30%

ACFE=15%

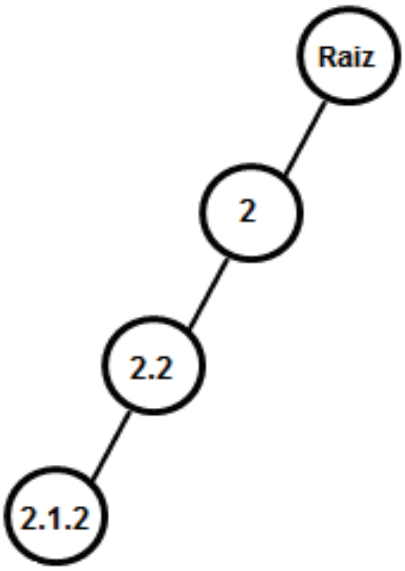

Figura 5.14 Classe da terceira regra depois da modificação.

Dessa forma, o parâmetro ganhador foi o ACEE. Como c=1 e m=1(valores hipotéticos), a classe dessa regra vai ser ter 1 nível a menos e o último nó remanescente vai ser modificado, conforme mostrado por meio da figura 5.15.

A classe da regra 3 teve 1 nível a menos, passando de 3 níveis para 2, passando a classe a ser 2.1. A partir daí a classe teve o último nível modificado, passando de 2.1 para 2.3. Assim, a classe da regra 3 foi modificada parcialmente, já que antes era 2.1.2 e passou a ser 2.3, como ilustrado na figura 5.15.

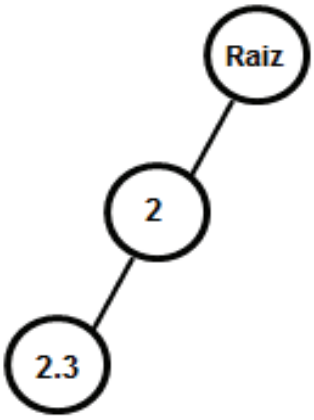

Figura 5.15 Classe da terceira regra depois da modificação.

A classe da quarta regra usada para demonstração do método e mostrada na figura 5.16.

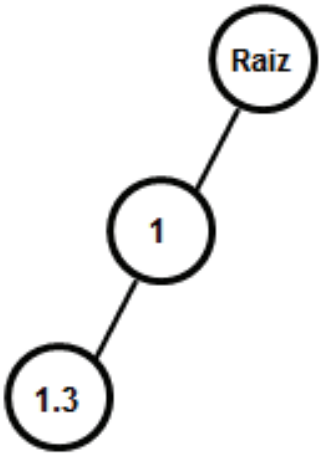

Figura 5.16 Classe da quarta regra antes da modificação.

A regra 4 teve as seguintes porcentagens dos parâmetros:

AT=5%

AP=10%

ACF=10%

ACE = 10%

ACEE=20%

ACFE=45%

Dessa forma, o parâmetro ganhador foi o ACFE. Como *d*=1 e *f*=1 (valores hipotéticos), a classe dessa regra vai ser ter o último nível modificado e 1 nível a mais.

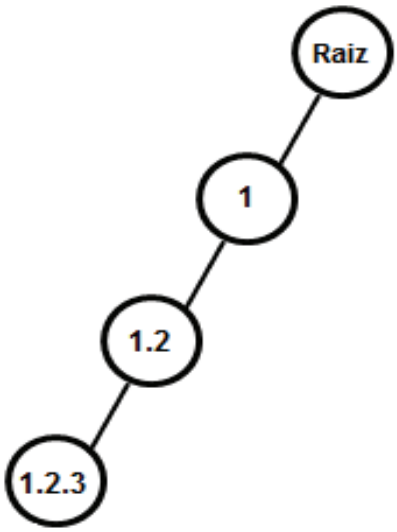

Figura 5.17 Classe da quarta regra depois da modificação.

A classe 1.3 teve, inicialmente, 1 nível modificado, passando de 1.3 para 1.2. A partir daí, teve 1 nível a mais, passando de 1.2 para 1.2.3. Portanto, conforme ilustrado na figura 17, a classe da regra 4 foi modificada parcialmente, já que era 1.3 e passou a ser 1.2.3.

5.2.2 Avaliação das regras

Para a avaliação de cada uma das regras (indivíduos), serão levados em conta apenas os exemplos que ela cobriu. Pelo mesmo motivo descrito na avaliação das regras do RCM-LST, no RCM-GLT as regras podem ser avaliadas positiva e negativamente. Assim, quando a classe prevista pela regra tiver ancestrais em comum com a classe de um exemplo que ela cobriu, a regra é avaliada positivamente, enquanto quando a classe prevista pela regra não tiver ancestrais em comum com a classe de um exemplo que ela cobriu, ela é avaliada negativamente. Portanto, as regras são avaliadas de acordo com a equação 5.19:

$$\boldsymbol{Fit_I = NCP/(1 + NCN)} \qquad (5.19)$$

Em que NCP é o numero de casos em que as classes prevista e correta possuem ancestrais em comum e NCN o número de casos em que as classes prevista e correta não possuem nenhum ancestral em comum.

O indivíduo 1 usado para exemplificar a forma de avaliação do método é representado pela seguinte regra: Se Atributo A = 2 e Atributo B = 0 e Atributo C = 3 então Classe = R.1.1. *numgenes1* = 2 *genes1*={Atributo A, Atributo B}. A base descrita na tabela 5.2 será usada para exemplificar o uso da regra.

Tabela 5.2 Descrição dos exemplos que compõem a base de testes para exemplificação do método usado para avaliação das regras no RCM-GST.

Exemplo/Atributo	A	B	C	Classe
1	0	1	1	1
2	1	3	3	2.3.1
3	2	0	2	1.1
4	2	0	0	2.2
5	2	0	3	1.2

A árvore usada para representação das classes usadas para exemplificação do método de avaliação das regras é mostrada na figura 5.18.

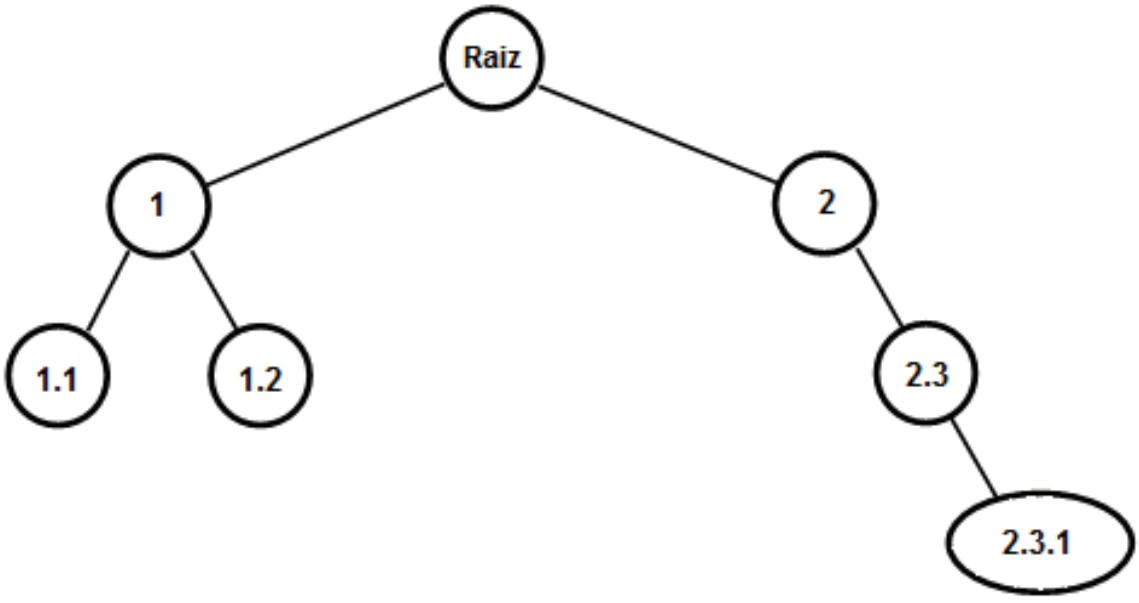

Figura 5.18 Ilustração da árvore de classes usada no exemplo do método de avaliação das regras.

De acordo com o dados apresentados, o indivíduo 1 cobre os exemplos 3, 4 e 5. Portanto, para o cálculo de Fit_I, Ecr=3, sendo que a classe prevista pelo indivíduo será comparada às classes dos exemplos 3,4 e 5.

NCP = 1, pois existem ancestrais em comum entre as classes prevista (1.1) e a classe do exemplo 3 (1.1).

NCN = 1, pois não existem ancestrais em comum entre as classes prevista (1.1) e a classe do exemplo 3 (2.2).

NCP = 2, pois existem ancestrais em comum entre as classes prevista (1.1) e a classe do exemplo 5 (1.2). Portanto,

$Fit_I = 2/(1+1)$

$Fit_I = 1$

Nas seções 5.1 e 5.2 foram descritos métodos local e global de rótulo simples com a estrutura de classes representada por uma árvore. Nas seções 5.3 e 5.4 são descritos os métodos de rótulo simples para uma hierarquia de classes representadas por um DAG.

5.3 RCM-LSD

O objetivo do RCM-LSD é a extração de regras de classificação de proteínas para a formação um classificador hierárquico local de rótulo simples para uma hierarquia de classes representada por um DAG. O RCM-LSD usa a abordagem local por nó. Assim, o método constrói um classificador para cada nó.

A construção de todos os itens de cada classificador com o uso do RCM-LSD é feita da mesma maneira como no uso do RCM-LST. Porém, como no RCM-LSD os classificadores são binários, a avaliação das regras no RCM-LSD é feita de maneira idêntica à feita no RCM-LST. O método RCM-GSD, método que se diferencia do RCM-LSD por ser global e não local, é descrito na seção 5.4.

5.4 RCM-GSD

O objetivo do RCM-GSD é semelhante ao do RCM-LSD. Porém, o RCM-GSD gera classificadores globais A construção do classificador com o uso do RCM-GSD é feita da mesma maneira como no uso do RCM-GST, com exceção da

avaliação das regras, que no RCM-GSD é feita como descrito na seção 5.4.1. No RCM-GSD a *Fi* é calculada da mesma forma como feito no RCM-GST. Os métodos multirrótulo são descritos na seção de 5.5 a 5.8, sendo que na seção 5.5 é descrito o método local para uma hierarquia de classes representada por um grafo.

5.5 RCM-LMT

O RCM-LMT gera regras de classificação de proteínas que formam classificadores hierárquicos locais multirrótulo. No RCM-LMT, hierarquia de classes é representada por uma árvore. O RCM-LMT é construído com o uso da abordagem local por nó. A construção do classificador com o uso do RCM-LMT é feita da mesma maneira como no uso do RCM-LST, com exceção da avaliação das regras, que no RCM-LMT é feita como descrito na seção 5.5.1.

5.5.1 Avaliação de regras

A avaliação de uma regra *i* no RCM-LMT é feita de acordo com a equação 5.20.

$$\boldsymbol{Fit_I = NCP/(1 + NCN)} \qquad (5.20)$$

Em que NCP é o numero de casos em que a classe prevista é igual a ao menos uma das classes corretas. NCN é o número de casos em que a classe prevista é diferente de todas as classes corretas. Na seção 5.6 é descrito o método, RCM-GMT, semelhante ao RCM-LMT, porém global.

5.6 RCM-GMT

O RCM-GMT extrai regras de classificação de proteínas a partir de uma base de dados com exemplos já rotulados com o objetivo de formar um classificador hierárquico global multirrótulo para uma hierarquia de classes representada por uma árvore. A construção do classificador com o uso do RCM-GMT é feita da mesma maneira como no uso do RCM-GST, com exceção da avaliação das regras, que no RCM-GMT é feita como descrito na seção 5.6.1.

5.6.1 Avaliação de regras

A avaliação de uma regra *i* no RCM-GMT é feita de acordo com a equação 5.21.

$$\boldsymbol{Fit_I = NCP/(1 + NCN)} \quad (5.21)$$

Em que NCP é o numero de casos em que a classe prevista e ao menos uma das classes corretas possuem ancestrais em comum. NCN é o número de casos em que a classe prevista não possui nenhum ancestral em comum com nenhuma das classes corretas. Nas seções 5.7 e 5.8 são descritos os métodos globais multirrótulo para uma hierarquia de classes representada por um DAG.

5.7 RCM-LMD

O objetivo do RCM-LMD é a construção de classificadores hierárquicos locais multirrótulo da função de proteínas por meio da extração de regras de classificação a partir de uma base de dados com exemplos já rotulados. A hierarquia entre as classes no RCM-LMD é representada por um DAG. O RCM-LMD é construído com o uso da abordagem local por nó.

A construção do classificador com o uso do RCM-LMD é feita da mesma maneira como no uso do RCM-LSD, com exceção da avaliação das regras, que no RCM-LMD é feita como descrito na seção 5.7.1. A avaliação de uma regra *i* no RCM-MT é feita de maneira idêntica à feita no RCM-LST.

Na seção 5.8 é descrito, RCM, método que cuja construção é o principal objetivo deste trabalho. Os métodos anteriores foram criados com o objetivo de se desenvolver o RCM com o menor número de limitações possível.

5.8 RCM

O objetivo do RCM é a construção de classificadores hierárquicos globais das funções de proteínas a partir de uma base de dados com exemplos já rotulados. As regras extraídas formam um classificador para uma hierarquia de classes representada por um DAG. A construção do classificador com o uso do RCM é feita da mesma maneira como no uso do RCM-GSD.A avaliação de uma regra *i* no RCM é feita de maneira idêntica à feita no RCM-GMT.

Capítulo 6

Experimentos

Neste capítulo são descritos os experimentos feitos com os métodos descritos neste trabalho. Os experimentos foram divididos em quatro etapas para facilitar o entendimento da descrição dos experimentos e dos resultados.

O RCM é o único método disponível na literatura que constrói classificadores hierárquicos de maneira independente das característica do problema apresentado. Dessa forma, para cada versão do RCM foi escolhido um método de mesmo objetivo para ser comparado. Em alguns casos, foi usado o mesmo método para a comparação com diferentes versões do RCM. Em todas as etapas os algoritmos que usaram um dos métodos RCM foram executados usando os valores descritos na tabela 6.1.

Tabela 6.1 Valores das variáveis usadas para a execução dos algoritmos descritos neste trabalho nos experimentos em todas as etapas.

Variável	**Valor**
min (número mínimo de genes para um indivíduo)	5
max (número máximo de genes para um indivíduo)	9
N (número de indivíduos)	70
G (número de gerações)	100

De acordo com [Miller, 1956], o ser humano é capaz de compreender até 7 pedaços de informação. Dessa forma, foram geradas regras com 7±2 condições. [Das et al., 2009] sugere que o número de indivíduos deve ser 10 vezes o número de genes por indivíduo. Como foram gerados indivíduos com 7±2 genes, definiu-se que o número de indivíduos por geração é 7*10. Experimentos preliminares foram executados com todas as bases com o objetivo de se definir o número de gerações (G). Foram executados experimentos com 50, 10 e 200 gerações. Baseado nos experimentos preliminares foi escolhido G = 100.

Os algoritmos das etapas 1 e 2 foram executados usando validação cruzada fator 10. Os algoritmos das etapas de 3 e 4 foram executados 10 vezes usando o método *holdout*. Em todos os casos, o método foi escolhido por ter sido usado nos experimentos do algoritmo que está sendo comparado com o RCM. Os experimentos feitos em cada uma das quatro etapas são descritos nas seções de 6.1 a 6.4.

6.1 Etapa 1

Na primeira etapa dos experimentos, inicialmente, o RCM-LST foi comparado ao RCM-GST. Posteriormente, o RCM-LST foi comparado ao LMNBwU e o RCM-GST comparado ao GMNBwU e ao HLCS-Tree. As bases usadas em todos os experimentos desta etapa são as mesmas usadas nos experimentos feitos com os métodos GMNBwU e LMNBwU em [Silla and Freitas, 2009] (Tabela 5.2). Foram usadas quatro bases com exemplos de proteínas GPCR e quatro bases com exemplos de enzimas. Nesta etapa, os métodos foram avaliados com o uso das medidas hP, hR e hF, descritas nas equações 3.4, 3.6 3.8, respectivamente.

Tabela 6.2 Características das bases usadas nos experimentos usando o RCM-LST e LMNBwU.

Tipo	Nome	Nº de atributos	Nº de exemplos	Nº de classes
GPCR	Pfam	75	7053	192
	Prints	283	5404	179
	Prosite	129	6256	187
	Interpro	450	7444	198
Enzimas	Pfam	708	13987	333
	Prints	352	14025	351
	Prosite	585	14041	324
	Interpro	1216	14027	330

As bases de dados descritas na tabela 6.2 são as mesmas usadas em [Holden and Freitas, 2008a] e [Silla and Freitas, 2009]. Essas bases foram criadas especificamente para o uso em [Holden and Freitas, 2008a] e foram disponibilizadas para os autores deste trabalho pelos autores de [Silla and Freitas, 2009]. Neste

trabalho foram realizados experimentos com outras versões do RCM e com outras bases de dados. Uma parte desses experimentos é descrita na seção 6.2.

6.2 Etapa 2

Na segunda etapa, inicialmente, o RCM-LSD foi comparado ao RCM-GSD. Posteriormente, o método RCM-GSD foi comparado *ao h-AntMiner* e ao HLCS-DAG. Para os experimentos descritos nesta seção, foram usadas cinco bases de dados, conforme descrito na tabela 5.3. As bases foram usadas nos experimentos do método *h-AntMiner* [Otero et al., 2009], sendo que elas são divididas em 2 grupos (Tabela 6.3).

Tabela 6.3 Características das bases usadas nos experimentos usando o RCM-LSD e RCM-GSD.

Grupo	Nome	Nº de atributos	Nº de instâncias	Nº de classes
ds1	aa	22	177	19
	Interpro	92	177	19
	IntAct	2096	147	19
ds2	aa	22	1631	17
	Interpro	155	478	17

Para os experimentos com o HLCS-DAG foram usadas as bases AA e Interpro de ambos os grupos. Os resultados dos experimentos a partir da base ds1 IntAct foram comparados apenas com os resultados do *h-AntMiner*. As bases de dados usadas nos experimentos desta etapa armazenam informações sobre canais iônicos, proteína transmembranar presente em todas as células vivas [Otero et al., 2009]. Essas bases foram criadas para serem usadas nos experimentos descritos em [Otero et al., 2009] e foram disponibilizadas para os autores deste trabalho pelos autores de [Otero et al., 2009]. Nesta etapa, os métodos foram avaliados com o uso das medidas hP, hR e hF. Além dos experimentos descritos nas seções 6.1 e 6.2 feito com o uso das versões de rótulo simples do RCM, foram feitos experimentos com o uso das versões multirrótulo do método, descritos na seção 6.3 (RCM-LMT e RCM-GMT) e 6.4 (RCM-LMD e RCM-GMD).

6.3 Etapa 3

Nos experimentos do HLCS-Multi, feitos em [Romão, 2012], para a classificação de um exemplo durante a fase de testes, o número de classes escolhidas pelo classificador para se rotular um exemplo é o mesmo numero de classes as quais o exemplo pertence. As classes escolhidas pelo classificador são as classes das regras que cobrem o exemplo a ser classificado. Caso o número de regras que cobrem o exemplo seja menor do que o número de classes as quais o exemplo pertence, o classificador escolhe novas classes para que o número de classes previstas seja igual ao número de classes corretas. No HLCS-Multi, as novas classes escolhidas são aquelas mais presentes na base de dados. Além disso, no HLCS-Multi, para a contagem das medidas, a cada predição das classes de uma instância, é escolhida a melhor combinação entre classes previstas e corretas. A primeira versão do RCM-GMT, chamada aqui de RCM-GMT 1, escolhe as classes previstas de maneira idêntica ao HLCS-Multi.

Porém, no caso da necessidade da classificação de uma nova proteína a qual não tenha nenhuma classe definida, seguindo as definições citadas, o RCM-GMT 1 e o HLCS-Multi não teriam um número pré-definido de classes a serem escolhidas. Com o objetivo de tornar o sistema mais próximo de situações em que o seu é requerido para predições reais, e não apenas de testes, foi criada a segunda versão do RCM-GMT, chamada aqui de RCM-GMT 2. A segunda versão escolhe, para um determinado exemplo, as classes de todas as regras que cubram aquele exemplo e que o seu *fitness* seja maior ou igual a l. O valor de l é definido como a média entre os *fitness* de todas as regras que cobrem o exemplo a ser classificado. Assim, para a predição de cada classe correta, é escolhida a classe prevista mais próxima a ela, sendo que a classe prevista escolhida é retirada da lista de classes previstas. Caso todas as classes previstas tenham a distância máxima da classe correta, uma delas é sorteada para ser excluída da lista.

Inicialmente, o RCM-LMT foi comparado ao RCM-GMT 1. A forma de escolha das classes de um exemplo na fase de teste foi a mesma. Além disso, o RCM-GMT 1 foi comparado ao HLCS-Multi, e ao RCM-GMT 2, separadamente. As comparações listadas nesta seção até aqui foram feitas usando as medidas hR, hP, hF. O RCM-GMT 1 e o RCM-GMT 2 também foram comparados entre si com o uso da

curva PR e da área sob a curva PR. Também foram comparados entre si, por meio da curva PR, os métodos HLCS-Multi, RCM-GMT 1, RCM-GMT 2 e Clus-HMC. Como em todas as bases o HLCS-Multi não apresentou curvas PR e sim pontos, não foi possível se calcular a área sob a curva PR para esse método. Dessa forma, os métodos RCM-GMT 1, RCM-GMT 2 e Clus-HMC foram comparados com o uso da área sob a curva PR. As bases usadas nos experimentos são descritas na tabela 6.4.

Tabela 6.4 Características das bases usadas nos experimentos da etapa 3.

Nome	Nº de instâncias	Nº de atributos	Nº de classes
Cellcycle	3757	77	500
Church	2911	27	500
Derisi	3725	63	500
Eisen	2424	79	462
Expr	3779	551	500
Gasch1	3764	173	500
Gasch2	2930	52	500
Pheno	1591	69	456
Seq	3919	478	500
Spo	1703	80	500

As 10 bases usadas são versões modificadas das bases usadas em [Clare and King, 2003]. Essas bases foram modificadas para uso nos experimentos descritos em [Vens et al., 2008] e possuem instância de proteínas que pertencem a classes definidas segundo a ontologia FunCat. As bases estão disponíveis em http://dtai.cs.kuleuven.be/clus/hmcdatasets). Os experimentos usando os métodos globais de multirrótulo para uma hierarquia de classes representada por um DAG são descritos na seção 6.4.

6.4 Etapa 4

Para esta etapa, pelo mesmo motivo citado na seção 6.3, o RCM-GMD foi executado durante a fase de testes de duas formas distintas, sendo essas formas as

mesmas explicadas na seção anterior. Elas foram chamadas aqui de RCM-GMD 1 e RCM-GMD 2. Inicialmente, o RCM-LMD foi comparado ao RCM-GMD 2. Posteriormente, o RCM-GMD 1 foi comparado ao HLCS-Multi e ao RCM-GMD 2, separadamente. As comparações listadas nesta seção até aqui foram feitas usando as medidas hR, hP, hF. Assim, o RCM-GMD 1 e o RCM-GMD 2 também foram comparados entre si com o uso da curva PR e da área sob a curva PR. Também foram comparados, por meio da curva PR, os métodos HLCS-Multi, RCM-GMD 1, RCM-GMD 2 e Clus-HMC. Como, em todas as bases, o HLCS-Multi não apresentou curvas PR e sim pontos, não foi possível se calcular a área sob a curva PR desse método. Dessa forma, os métodos RCM-GMD 1, RCM-GMD 2 e Clus-HMC foram comparados com o uso da área sob a curva PR. As bases usadas nos experimentos são descritas na tabela 6.5.

Tabela 6.5 Características das bases usadas nos experimentos da etapa 4.

Nome	# de instâncias	# de atributos	# de classes
Cellcycle	3751	77	4126
Church	2905	27	4126
Derisi	3719	63	4120
Eisen	2418	79	3574
Expr	3773	551	4132
Gasch1	3758	173	4126
Gasch2	2923	52	4126
Pheno	1586	69	3128
Seq	3900	478	4134
Spo	3697	80	4120

Assim como as bases usadas na etapa 3, as 10 bases usadas nos experimentos desta seção e descritas na tabela 6.5 são versões modificadas da versão original usada em [Clare and King, 2003]. As versões modificadas também foram usadas em [Vens et al., 2008], possuem instância de proteínas que pertencem a classes definidas

segundo a ontologia GO e estão disponíveis em estão disponíveis em http://dtai.cs.kuleuven.be/clus/hmcdatasets)..

Capítulo 7
Resultados

Este capítulo foi dividido em cinco seções. Nas quatro primeiras são apresentados os resultados das quatro primeiro etapas dos experimentos. Na última etapa são descritas as conclusões de cada um dos experimentos realizados.

Usando os resultados aqui descritos foram realizados três tipos de teste estatístico, sendo eles o t de *student* [Guimarães, 2008], o de Mann-Whitney [Guimarães, 2008] e o de Kruskal Wallis. [Theodorsson-Norheim, 1986] O t de *student* foi realizado para se comparar os resultados a cada base nos experimentos onde dois métodos RCM foram comparados entre si, já que foram os únicos casos onde se teve acesso aos resultados de cada uma das 10 execuções dos experimentos. Nos outros casos, se teve acesso apenas à média entre as dez execuções. Assim, quando foi feita a comparação entre dois métodos e um dos métodos não é RCM, os testes estatísticos foram feitos para comparação entre os métodos levando-se em consideração todas as bases de uma só vez. Nos casos em que não foi usado um dos métodos descritos neste trabalho, foram considerados semelhantes os experimentos com diferença entre os resultados menor ou igual a 3%. Esse valor foi escolhido porque, usando o teste t de student, se a diferença média entre as dez execuções feitas entre dois métodos for menor ou igual a três, não há diferença estatisticamente significativa entre os resultados. Se a diferença for maior do que 3%, a diferença é estatisticamente significante. Os testes de Mann-Whitney e Kruskal Wallis foram usados para se comparar dois métodos levando em consideração os experimentos realizados a partir de todas as bases. Foi usado 95% de grau de confiança em todos os testes estatísticos.

7.1 Etapa 1

Os resultados dos experimentos do RCM-LST e do RCM-GST são apresentados na tabela 7.1. Os resultados em que houve diferença estatisticamente significante estão marcados em negrito na linha correspondente ao método que obteve vantagem. Como pode ser observado, usando as bases com proteínas GPCR, os métodos RCM-LST e RCM-GST apresentaram resultados semelhantes,

independentemente da medida usada.A partir das bases com enzimas,o RCM-GST superou o RCM-LST com o uso de todas as medidas e das bases Pfam e Prosite, com o uso das medidas hR e hP e a base Prints e da medida hP com a base Interpro.

Tabela 7.1 Resultados dos experimentos dos métodos RCM-LST e RCM-GST usando as bases com exemplos de enzimas e proteínas GPCR.

GPCR					
Dataset	Measure	Pfam	Prints	Prosite	Interpro
RCM-LST	hR	43.72±0.09	56.22±0.04	42.79±0.03	64.33±0.04
	hP	42.53±0.10	61.82±0.03	43.44±0.07	62.37±0.03
	hF	43.07±0.09	58.86±0.03	43.11±0.03	63.33±0.03
RCM-GST	hR	48.59±0.02	60.57±0.03	43.31±0.03	63.82±0.05
	hP	47.47±0.02	59.25±0.04	42.10±0.03	63.24±0.05
	hF	48.02±0.02	59.90±0.04	42.69±0.03	63.52±0.05
Enzimas					
Dataset	Measure	Pfam	Prints	Prosite	Interpro
RCM-LST	hR(%)	84.58±0.04	83.10±0.04	86.70±0.03	89.64±0.03
	hP(%)	79.20±0.04	78.03±0.04	83.60±0.02	84.13±0.03
	hF(%)	80.80±0,04	80.48±0.04	85.12±0.03	86.80±0.03
RCM-GST	hR(%)	**88.08±0.04**	**85.55±0.03**	**89.30±0.02**	89.22±0.03
	hP(%)	**88.31±0.04**	**84.43±0.02**	**88.98±0.03**	**88.69±0.03**
	hF(%)	**88.19±0.04**	84.98±0.02	**89.14±0.03**	88.95±0.03

Usando o teste de Mann-Whitney se chegou à conclusão de que a diferença entre os dois métodos é estatisticamente insignificante, independente da medida usada. Os resultados dos experimentos do RCM-LST e do LMNBwU são apresentados na tabela 7.2. Os resultados em que o RCM-LST superou ou obteve resultados semelhantes ao LMNBwU estão marcados em negrito. Como se pode observar, usando as bases com proteínas GPCR, o RCM-LST superou o LMNBwU

usando apenas uma das medidas (hR) e uma das bases(Interpro). Além disso, com o uso das bases contendo exemplos de enzimas, o RCM-LST superou ou obteve resultados semelhantes em todos os experimentos. Com o uso das bases Pfam e Interpro, com qualquer uma das medidas, o RCM-LST obteve melhores resultados do que o LMNBwU. Com as bases Prints e Prosite, no uso de qualquer uma três medidas, os dois algoritmos obtiveram resultados semelhantes. A diferença entre resultados dos dois algoritmos foi estatisticamente insignificante (teste de Mann-Whitney).

Tabela 7.2 Resultados dos experimentos dos métodos RCM-LST e LMNBwU usando as bases com exemplos de enzimas e proteínas GPCR.

Tipo	Nome	RCM-LST			LMNBwU		
		hR(%)	hP(%)	hF(%)	hR(%)	hP(%)	hF(%)
GPCR	Pfam	43.72±0.09	42.53±0.10	43.07±0.09	59.17	66.49	61.32
	Prints	56.22±0.04	61.82±0.03	58.86±0.03	66.32	70.13	66.99
	Prosite	42.79±0.03	43.44±0.07	43.11±0.03	55.95	63.45	58.11
	Interpro	**64.33±0.04**	62.37±0.03	63.33±0.03	67.29	70.49	67.90
Enzimas	Pfam	**84.58±0.04**	**79.20±0.04**	**80.80±0,04**	79.73	74.94	76.47
	Prints	**83.10±0.04**	**78.03±0.04**	**80.48±0.04**	82.73	78.35	79.79
	Prosite	**86.70±0.03**	**83.60±0.02**	**85.12±0.03**	86.52	81.73	83.20
	Interpro	**89.64±0.03**	**84.13±0.03**	**86.80±0.03**	80.23	74.85	76.64

Os resultados dos experimentos com a utilização dos métodos RCM-GST, GMNBwU o HLCS-Tree a partir das bases com exemplos de proteínas GPCR são apresentados na tabela 7.3. São marcados em negrito os resultados do RCM-GST semelhantes ou superiores aos resultados de pelo menos um dos outros dois algoritmos. Como pode ser observado, usando as bases GPCR, o RCM-GST foi superado pelos outros dois algoritmos, independente da medida usada. Além disso, os resultados apresentados na tabela 7.3 mostram que o RCM-GST mostrou resultados superiores com o uso das duas bases com o maior número de atributos (283 e 450, para as bases Prints e Interpro, respectivamente). Os outros dois algoritmos usados também apresentaram resultados superiores com o uso das bases Prints e Interpro.

Tabela 7.3 Resultados dos experimentos dos métodos RCM-GST, GMNBwU e HLCS-Tree usando as bases com exemplos de proteínas GPCR.

GPCR					
Dataset	Measure	Pfam	Prints	Prosite	Interpro
RCM-GST	hR	46.79±0.04	60.40±0.02	40.75±0.04	62.65±0.05
	hP	46.42±0.05	58.94±0.02	39.70±0.04	62.40±0.06
	hF	46.60±0.04	59.66±0.02	40.21±0.04	62.52±0.05
GMNBwU	hR	57.52	69.42	53.73	71.33
	hP	77.23	87.06	75.64	87.60
	hF	64.40	75.38	61.14	77.01
HLCS-Tree	hR	60.30	68.18	60.45	74.30
	hP	82.53	86.50	79.42	90.26
	hF	69.69	76.25	68.65	81.51

Os resultados dos experimentos do RCM-GST, do GMNBwU e o HLCS-Tree a partir das bases com exemplos de proteínas GPCR são apresentados na tabela 7.3. São marcados em negrito os resultados do RCM-GST semelhantes ou superiores aos resultados de pelo menos um dos outros dois algoritmos. Usando as tabelas com exemplos de enzimas, o método RCM-GST só não obteve resultados semelhantes ou superiores a pelo menos um dos outros dois algoritmos em um dos experimentos. Com o uso da base Pfam e as medidas hR e hF, o RCM-GST apresentou resultados semelhantes ao GMNBwU e superior ao HLCS-Tree. Também usando a base Pfam, com a medida hP, o RCM-GST apresentou resultados semelhantes ao HLCS-Tree. Usando a base Prints e a medida hP, o RCM-GST apresentou resultados semelhantes aos outros dois algoritmos. O RCM-GST superou o HLCS-Tree e obteve resultados semelhantes ao GMNBwU usando a base Prints e a medida hR. Usando a mesma base e a medida hF, o RCM-GST apresentou resultados semelhantes ao HLCS-Tree. Apenas no uso da base Prints e a medida hP o RCM-GST foi superado pelos outros dois algoritmos.

Tabela 7.4 Resultados dos experimentos dos métodos RCM-GST, GMNBwU e HLCS-Tree usando as bases com exemplos de enzimas.

Enzimas					
Dataset	Measure	Pfam	Prints	Prosite	Interpro
RCM-GST	hR(%)	**86.04±0.12**	**84.53±0.03**	**89.85±0.02**	**89.58±0.03**
	hP(%)	**84.65±0.13**	83.37±0.03	**89.60±0.02**	**89.21±0.03**
	hF(%)	**85.34±0.13**	**83.94±0.03**	**89.72±0.02**	**89.39±0.03**
GMNBwU	hR(%)	86.94	87.26	89.53	89.58
	hP(%)	95.15	92.21	95.14	94.96
	hF(%)	88.72	87.98	90.70	90.53
HLCS-Tree	hR(%)	81.47	82.33	86.27	85.36
	hP(%)	86.34	89.69	90.35	87.80
	hF(%)	83.83	85.85	88.26	86.56

A partir da base Prosite, o RMCDE-GST superou o HLCS-Tree usando a medida hR e obteve resultados semelhantes aos dois algoritmos usando as medidas hP e hF. Com o uso da base Interpro, o RCM-GST superou o HLCS-Tree com o uso das medidas hR e obteve resultados semelhantes usando as medidas hF e hP. Usando as medidas hR e hF, o RCM-GST obteve resultados semelhantes ao GMNBwU. Usando o teste de Kruskall Wallis, concluiu-se que a diferença entre os resultados dos três métodos é estatisticamente insignificante.

Na tabela 7.5 estão o número de regras geradas e a média de regras geradas por classe pelo RCM-GST a partir de cada uma das bases. Em ambos os casos foram calculadas as médias entre os dez experimentos realizados. Como pode ser observado, o número de instâncias na base influencia no número de regras geradas, já que as quatro bases com o maior número de exemplos (as quatro bases que armazenam enzimas-grupo EC) são aquelas com o maior número de regras. Sugere-se que o número de atributos também influencia no número de regras geradas, já que a lista de bases ordenadas de maneira crescente com base no número de atributos é a mesma lista de bases ordenadas de maneira crescente com base no número de regras geradas.

Tabela 7.5 Resultados dos experimentos dos métodos RCM-GST, GMNBwU e HLCS-Tree usando as bases com exemplos de proteínas GPCR.

Base	Número de regras	Média de regras por classe
GPCRPfam	64,7±1.,84	1,30±0,11
GPCRPrints	100±7,81	1,38±0,06
GPCRProsite	89,2±10,38	1,72±0,11
GPCRInterpro	151,4±6,85	1,63±0,05
ECPfam	454,9±20,15	2,13±0,08
ECPrints	305,1±9,69	1,61±0,05
ECProsite	309,6±12,19	1,62±0,07
ECInterpro	472,9±13,55	2,21±0,06

Os resultados da segunda etapa dos experimentos são descritos a seguir.

7.2 Etapa 2

Os resultados dos experimentos com o método RCM-LSD e RCM-GSD são descritos na tabela 7.6 e 7.7, respectivamente.

Tabela 7.6 Resultados dos experimentos do método RCM-LSD usando as bases com exemplos de canais iônicos.

RCM-LSD				
		hR	hP	hF
ds1	aa	0.30±0.10	0.54±0.16	0.38±0.11
	Interpro	0.51±0.05	**0.73±0.06**	0.60±0.05
	IntAct	0.60±0.02	**0.79±0.04**	0.68±0.02
ds2	aa	0.32±0.10	0.50±0.07	0.39±0.06
	Interpro	0.50±0.09	0.59±0.08	0.54±0.07

Tabela 7.7 Resultados dos experimentos do método RCM-GSD usando as bases com exemplos de canais iônicos.

RCM-GSD				
		hR	hP	hF
ds1	AA	**0.57±0.12**	0.56±0.12	0.56±0.09
	Interpro	0.83±0.05	0.58±0.05	0.68±0.04
	IntAct	**0.88±0.02**	0.67±0.03	**0.74±0.02**
ds2	AA	**0.63±0.04**	0.62±0.04	**0.62±0.03**
	Interpro	0.58±0.02	**0.81±0.02**	0.68±0.02

São marcados em negrito os resultados em que um método superou o outro. O método RCM-LSD superou o RCM-GSD com o uso das bases Interpro e Intact do grupo ds1 e da medida hP. A partir das bases Intact do grupo ds1 e aa do grupo ds2, o RCM-GSD superou o RCM- LSD usando as medidas hR e hF. Além disso, o método global superou o local usando a base aa do grupo ds1 e a medida hR e a base aa do grupo ds2 usando a medida hP. Não houve diferença estatisticamente significante entre os resultados. Para os experimentos apresentados na tabela 7.6 foi usado o teste estatístico de Mann-Whitney.

Os resultados dos experimentos usando os métodos RCM-GSD, hAntMiner e HLCS-DAG a partir de quatro das cinco bases com exemplos de canais iônicos são apresentados na tabela 7.7. Os resultados dos experimentos usando os métodos RCM-GSD e hAntMiner a partir de uma das cinco bases com exemplos de canais iônicos são apresentados na tabela 7.8. São marcados em negrito os resultados em que o método RCM-GSD superou ou obteve resultados semelhantes a pelo menos um dos outros dois métodos. Nos resultados apresentados na tabela 7.7, com o uso da medida hR, o RCM-GSD superou ou obteve resultados semelhantes aos outros dois algoritmos em todas as 4 bases . O RCM-GSD superou o HLCS usando a medida hP e a base Interpro do grupo ds2, usando a base ds1 Interpro e a medida hR, a base ds1 Interpro e ds2 Interpro e a medida hF. Quando a base aa do grupo ds2 e a medida hR foram usadas, o RCM-GSD superou os dois outros algoritmos usados.

Tabela 7.8 Resultados dos experimentos dos métodos RCM-GST e hAntMiner usando as bases com exemplos de canais iônicos.

RCM-GSD				
Grupo	Nome	hP	hR	hF
ds1	AA	**0.56±0.12**	**0.57±0.12**	**0.56±0.09**
	Interpro	**0.58±0.05**	**0.83±0.05**	**0.68±0.04**
ds2	AA	**0.62±0.04**	**0.63±0.04**	**0.62±0.03**
	Interpro	**0.81±0.02**	**0.58±0.02**	**0.68±0.02**
hAnt-Miner				
Grupo	Nome	hP	hR	hF
ds1	AA	0.56±0.06	0.55±0.06	0.56±0.06
	Interpro	0.82±0.04	0.81±0.04	0.81±0.04
ds2	AA	0.63±0.02	0.59±0.02	0.61±0.01
	Interpro	0.83±0.01	0.75±0.01	0.79±0.01
HLCS				
Grupo	Nome	hP	hR	hF
ds1	AA	0.84±0.02	0.64±0.03	0.73±0.02
	Interpro	0.54±0.03	0.65±0.02	0.59±0.02
ds2	AA	0.86±0.04	0.58±0.03	0.69±0.01
	Interpro	0.64±0.04	0.61±0.03	0.63±0.02

Tabela 7.9 Resultados dos experimentos dos métodos RCM-GST e HLCS-DAG usando a base ds1 IntAct.

RCM-GSD				
Grupo	Nome	hP	hR	hF
ds1	IntAct	0.67±0.03	**0.88±0.02**	**0.74±0.02**
hAntMiner				
Grupo	Nome	hP	hR	hF
ds1	IntAct	0.77±0.04	0.54±0.03	0.63±0.03

Nos resultados apresentados na tabela 7.8, o RCM-GSD superou o hAntMiner com o uso da medida hR a partir da base ds1 IntAct e com o uso das medidas hR e hF. Não houve diferença significativa entre os resultados (teste de Kruskall Wallis). O número de regras geradas pelo RCM-GSD e a média do número de regras geradas por classe são mostrados na tabela 7.9. Nos dois casos foram calculadas as médias entre os dez experimentos realizados.

Tabela 7.10 Número de regras geradas e média de regras geradas por classe a partir de todas as bases.

Grupo	Nome	Número de regras	Média de regras por classe
ds1	AA	14,3±1,57	2,37±0,56
	Interpro	2±0	1±0
	IntAct	2,7±0,48	1,3±0,26
ds2	AA	22,6±3,34	2,71±0,62
	Interpro	2±0	1±0

Usando os resultados apresentados na tabela 7.9, sugere-se que o número de instâncias presentes na base de dados tem influência no número de regras geradas, já que a base com o maior número de exemplos (ds2AA) é a que tem o maior número de regras geradas. Quando as bases possuem um número menor de exemplos (ds2Interpro, ds1AA, ds1Interpro e ds1IntAct), o número de instâncias presentes na base e o número de atributos não se mostraram significativos na definição o número de regras. O resultados dos experimentos da etapa 3 são descritos na seção 7.3.

7.3 Etapa 3

Os resultados dos experimentos do RCM-LMT e do RCM-GMT 1 a partir das dez bases usadas são apresentados na tabela 7.10. São marcados em negrito os resultados em que um método superou o outro. Independentemente da medida e base usadas, o RCM-GMT 1 superou o RCM-LMT. Usando o teste de Mann-Whitney, conclui-se que o RCM-GMT1 teve desempenho estatisticamente superior ao RCM-LMT.

Tabela 7.11 Resultados dos experimentos dos métodos RCM-LMT e RCM-GMT.

RCM-LMT				RCM-GMT 1		
Nome	hP	hR	hF	hP	hR	hF
Cellcycle	0.0953±0.01	0.1046±0.01	0.0997±0.01	**0.1291±0.00**	**0.1289±0.00**	**0.1290±0.00**
Church	0.0950±0.01	0.1043±0.01	0.0994±0.01	**0.1245±0.00**	**0.1247±0.00**	**0.1245±0.00**
Derisi	0.0954±0.01	0.0947±0.01	0.0998±0.01	**0.1256±0.01**	**0.1269±0.01**	**0.1262±0.01**
Eisen	0.0909±0.01	0.0984±0.01	0.0945±0.01	**0.1357±0.00**	**0.1365±0.00**	**0.1361±0.00**
Expr	0.0949±0.01	0.1042±0.01	0.0994±0.01	**0.1382±0.00**	**0.1371±0.00**	**0.1376±0.00**
Gasch1	0.0951±0.01	0.1045±0.01	0.0995±0.01	**0.1429±0.01**	**0.1436±0.01**	**0.1432±0.01**
Gasch2	0.0951±0.01	0.1043±0.01	0.0995±0.01	**0.1371±0.01**	**0.1364±0.01**	**0.1367±0.01**
Pheno	0.0979±0.01	0.1078±0.01	0.1026±0.01	**0.1361±0.01**	**0.1385±0.01**	**0.1372±0.01**
Seq	0.0937±0.01	0.1028±0.01	0.0980±0.01	**0.1331±0.01**	**0.1317±0.01**	**0.1324±0.01**
Spo	0.0956±0.01	0.1048±0.01	0.1000±0.01	**0.1287±0.01**	**0.1307±0.01**	**0.1297±0.01**

Os resultados dos experimentos do RCM-GMT 1 contra o HLCS-Multi a partir das dez bases usadas são apresentados na tabela 7.11 e 7.12. São marcados em negrito os resultados em que o RCM-GMT 1 superou o HLCS-Multi ou em que os dois obtiveram resultados semelhantes.

Tabela 7.12 Resultados dos experimentos dos métodos RCM-GMT 1 e HLCS-Multi usando cinco das dez bases.

RCM-GMT 1				**HLCS-Multi**		
Nome	**hP**	**hR**	**hF**	**hP**	**hR**	**hF**
Cellcycle	0.1291±0.00	0.1289±0.00	0.1290±0.00	0.2777±0.04	0.2496±0.05	0.2629±0.04
Church	0.1245±0.00	0.1247±0.00	0.1245±0.00	0.2601±0.03	0.2059±0.03	0.2298±0.03
Derisi	0.1256±0.01	0.1269±0.01	0.1262±0.01	0.3200±0.03	0.3440±0.02	0.3316±0.03
Eisen	0.1357±0.00	0.1365±0.00	0.1361±0.00	0.2463±0.04	0.2096±0.05	0.2265±0.05
Expr	0.1382±0.00	0.1371±0.00	0.1376±0.00	0.2243±0.04	0.1886±0.03	0.2049±0.03

Tabela 7.13 Resultados dos experimentos dos métodos RCM-GMT 1 e HLCS-Multi usando cinco das dez bases.

	RCM-GMT 1			HLCS-Multi		
Nome	**hP**	**hR**	**hF**	**hP**	**hR**	**hF**
Gasch1	0.1429±0.01	0.1436±0.01	0.1432±0.01	0.2384±0.02	0.2048±0.01	0.2203±0.02
Gasch2	0.1371±0.01	**0.1364±0.01**	0.1367±0.01	0.2427±0.05	0.1611±005	0.1936±0.04
Pheno	0.1361±0.01	0.1385±0.01	0.1372±0.01	0.3082±0.02	0.5686±0.04	0.3997±0.03
Seq	0.1331±0.01	0.1317±0.01	0.1324±0.01	0.2320±0.02	0.1992±0.03	0.2147±0.03
Spo	0.1287±0.01	0.1307±0.01	0.1297±0.01	0.2466±0.03	0.1869±0.03	0.2126±0.02

O RCM-GMT 1 apresentou resultado semelhante ao HLCS-Multi apenas com o uso da medida hP a partir da base gasch 2. Assim, usando as medidas hR, hP e hF o HLCS-Multi foi superior, com diferença estatisticamente significante, ao RCM-GMT 1. Os experimentos usando os métodos do RCM-GMT 1 e RCM-GMT 1 têm seus resultados a partir das dez bases usadas são apresentados nas tabelas 7.13 e 7.14. São marcados em negrito os resultados em que o RCM-GMT 1 superou o RCM-GMT 2 ou em que os dois obtiveram resultados semelhantes.

Tabela 7.14 Resultados dos experimentos dos métodos RCM-GMT 1 e RCM-GMT 2 a partir de cinco das dez bases.

	RCM-GMT 1			RCM-GMT 2		
Nome	**hP**	**hR**	**hF**	**hP**	**hR**	**hF**
Cellcycle	**0.1291±0.00**	**0.1289±0.00**	**0.1290±0.00**	0.0997±0.01	0.0994±0.01	0.0995±0.01
Church	**0.1245±0.00**	**0.1247±0.00**	**0.1245±0.00**	0.0828±0.02	0.0834±0.02	0.0830±0.02
Derisi	**0.1256±0.01**	**0.1269±0.01**	**0.1262±0.01**	0.0681±0.01	0.0679±0.01	0.0680±0.01
Eisen	**0.1357±0.00**	**0.1365±0.00**	**0.1361±0.00**	0.0684±0.01	0.0689±0.01	0.0686±0.01
Expr	0.1382±0.00	0.1371±0.00	0.1376±0.00	**0.2581±0.01**	**0.2486±0.01**	**0.2503±0.01**

Com exceção dos experimentos feitos usando a base Expr, o RCM-GMT 1 superou o RCM-GMT2 em todos os experimentos.

Tabela 7.15 Resultados dos experimentos dos métodos RCM-GMT 1 e RCM-GMT 2 a partir de cinco das dez bases.

	RCM-GMT 1			RCM-GMT 2		
Nome	**hP**	**hR**	**hF**	**hP**	**hR**	**hF**
Gasch1	**0.1429±0.01**	**0.1436±0.01**	**0.1432±0.01**	0.0861±0.01	0.0844±0.01	0.0852±0.01
Gasch2	**0.1371±0.01**	**0.1364±0.01**	**0.1367±0.01**	0.0955±0.01	0.0940±0.01	0.0957±0.01
Pheno	**0.1361±0.01**	**0.1385±0.01**	**0.1372±0.01**	0.0763±0.01	0.0768±0.02	0.0763±0.01
Seq	0.1331±0.01	0.1317±0.01	0.1324±0.01	**0.1707±0.02**	**0.1694±0.02**	**0.1700±0.02**
Spo	**0.1287±0.01**	**0.1307±0.01**	**0.1297±0.01**	0.0792±0.01	0.0801±0.01	0.0796±0.01

Como se pode observar, o RCM-GMT 1 superou o RCM-GMT 2 a partir de oito das dez bases com o uso das três medidas. Além disso, o RCM-GMT 1 apresentou alguns resultados com desvio padrão iguais a zero (a partir das bases cellcyle, eisen e expr usando qualquer uma das medidas). O RCM-GMT 2 superou o RCM-GMT 1 a partir de duas das dez bases (expr e seq) usando qualquer umas das medidas.

Na tabela 7.16 são apresentadas as médias de classes previstas pelos classificadores gerados pelos métodos RCM-GMT 1 e RCM-GMT 2. Nas três primeiras colunas são apresentadas as médias de cada método, separadas por base de dados e na quarta coluna são apresentadas as médias do número de classes as quais as instâncias pertencem. Para os métodos RCM-LMT e RCM-GMT 1 são levadas em consideração apenas as classes das regras que cobriram as instâncias. Assim, as classes escolhidas pelos classificadores quando o número de regras que cobrem a instância é menor do que o número de classes as quais a instância pertence é desconsiderada. O método RCM-GMT 2 previu, em média, mais classes do que o RCM-GMT 1 em nove das dez bases. O RCM-LMT previu menos classes do que o método global a partir de todas as bases e, consequentemente, em média. Sugere-se que o RCM-GMT 2 tenha previsto mais classes do que o RCM-GMT 1 porque o RCM-GMT 1 prevê, no máximo, o número de classes corretas da instância, enquanto o RCM-GMT 2 prevê todas as classes das regras com grau de confiança maior do que um determinado limiar.

Tabela 7.16 Média do número de classes escolhidas pelos métodos RCM-LMT, RCM-GMT 1, RCM-GMT 2 e média do número de classes as quais as instâncias pertencem.

	RCM-LMT 1	RCM-GMT 1	RCM-GMT 2	Instância
Cellcycle	0,05	3,12	5,89	3,27
Church	0,01	3,25	4,57	3,26
Derisi	0,01	3,05	3,44	3,28
Eisen	0,01	3,28	3,26	3,41
Expr	0,26	3,16	20,08	3,26
Gasch1	0,01	3,02	3,78	3,26
Gasch2	0,01	3,19	5,15	3,26
Pheno	0,02	3,37	4,21	3,38
Seq	0,02	3,16	7,18	3,20
Spo	0,04	3,20	3,89	3,28
Média	0,04	3,18	6,14	3,28

Na tabela 7.17 e 7.18 são apresentados o número de regras geradas (médias entre os dez experimentos) e a média de regras geradas por classe (médias entre os dez experimentos) pelos métodos RCM-GMT1 e RCM-GMT2. Como os dois métodos se diferenciam apenas na escolha das regras a serem usadas para se classificar os exemplos, as regras geradas são as mesmas e consequentemente o número de regras geradas por eles são as mesmos. Os métodos são referenciados na tabela como RCM-GMT.

Tabela 7.17 Número de classes geradas e média de classes geradas por classe a partir de todas as bases pelos métodos RCM-GMT 1 e RCM-GMT2.

Base	Número de regras	Média de regras por classe
Cellcycle	859,3±54,42	3,46±0,13
Church	176,1±22,46	1,74±0,08
Derisi	838,4±18,66	3,47±0,10

Tabela 7.18 Número de classes geradas e média de classes geradas por classe a partir de todas as bases pelos métodos RCM-GMT 1 e RCM-GMT2.

Eisen	636,3±69,18	3,02±0,23
Expr	2015±29,17	5,75±0,13
Gasch1	1279, 3±81,783	4,28±0,19
Gasch2	527,8±25,93	2,60±0,10
Pheno	41,9±3,84	1,27±0,08
Seq	1972,8±88,52	5,77±0,25
Spo	519,9±41,57	2,65±013

Sugere-se que o número de regras geradas pelo RCM-GMT foi influenciado pelo número de instâncias e de atributos da base de dados. Isso pode ser observado por meio do fato de as duas bases a partir das quais o RCM-GMT gerou mais regras foram as duas com maior número de instâncias e atributos (Expr e Seq).

Na figura 7.1 são apresentadas as curvas PR dos métodos RCM-GMT 1 e RCM-GMT 2 a partir de 6 bases.

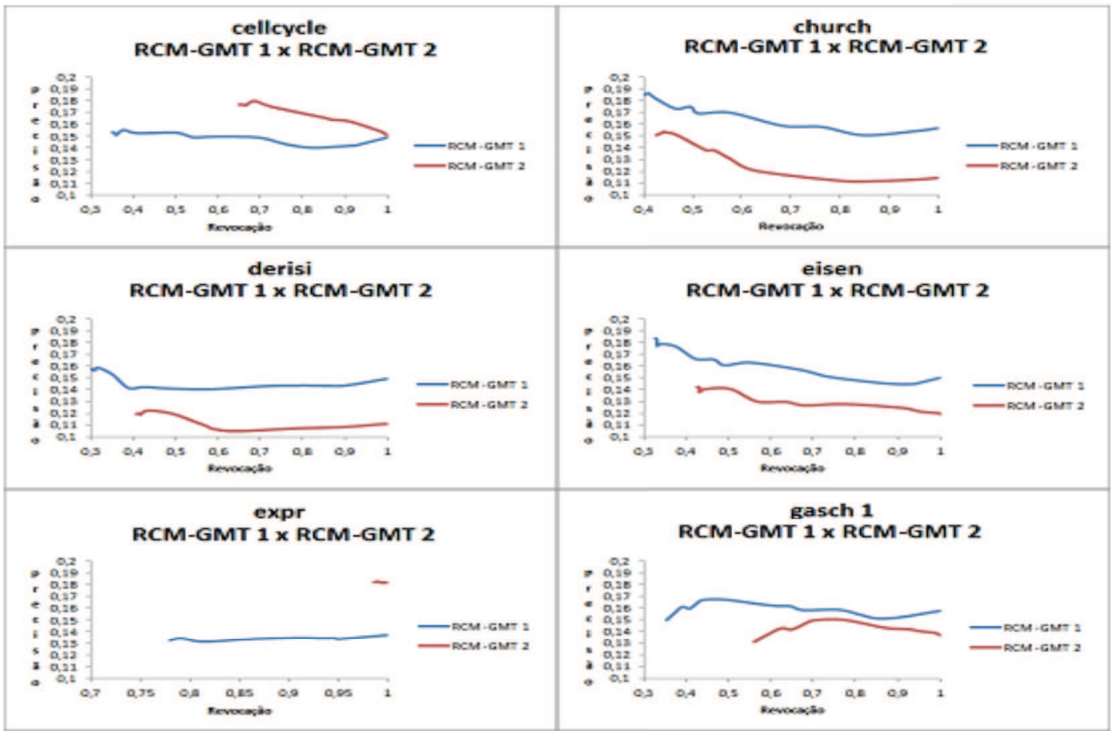

Figura 7.1 Curvas PR dos métodos RCM-GMT 1 e RCM-GMT2 a partir de seis das dez bases usadas.

Na figura 7.2 são apresentadas as curvas PR dos métodos RCM-GMT 1 e RCM-GMT 2 a partir de 6 bases.

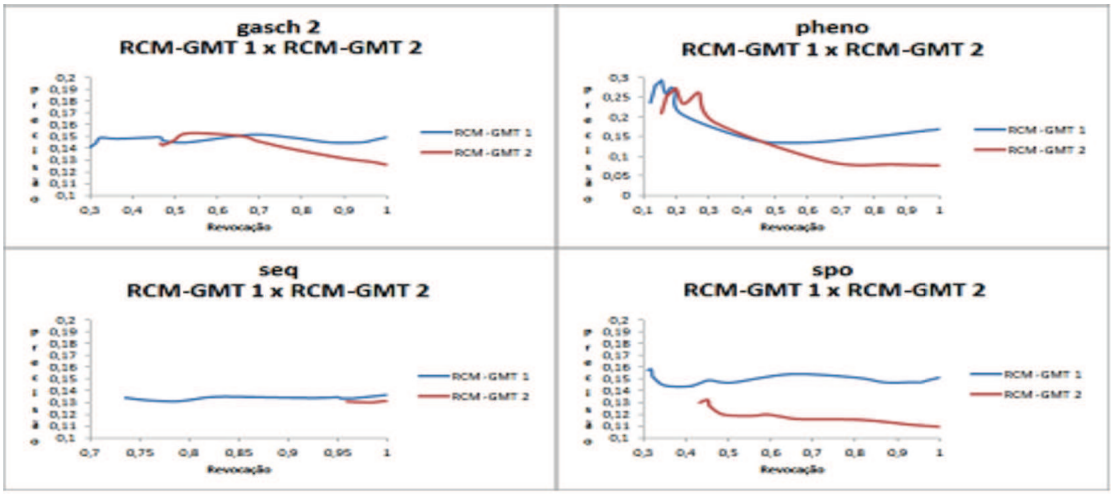

Figura 7.2 Curvas PR dos métodos RCM-G;T 1 e RCM-GMT 2 a partir de quatro das dez bases usadas.

Na maioria das curvas apresentadas nas figuras 7.1 e 7.2, os métodos apresentaram comportamentos parecidos, com as curvas do método RCM-GMT 1, em oito dos dez casos, se apresentando acima das curvas do método RCM-GMT 2.

Nas figuras 7.3 e 7.4 são apresentadas as curvas PR dos métodos HLCS-Multi, RCM-GMT 1, RCM-GMT 2 e Clus-HMC construídas com base nos experimentos feitos a partir de seis das dez bases descritas na tabela 7.4. O HLCS-Multi não apresentou curvas, e sim pontos, já que os resultados não variaram de acordo com os limiares.

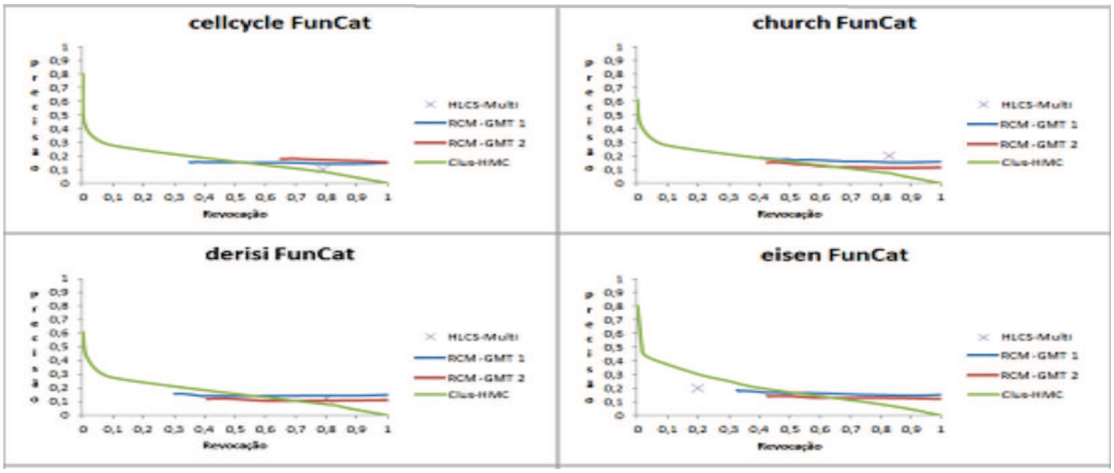

Figura 7.3 Curvas PR dos métodos RCM-GMT 1 e RCM-GMT 2 a partir de seis das dez bases descritas na tabela 7.4.

Na figura 7.4 são apresentadas as curvas PR dos quatro métodos construídas com base nos experimentos feitos a partir de quatro das dez bases usadas.

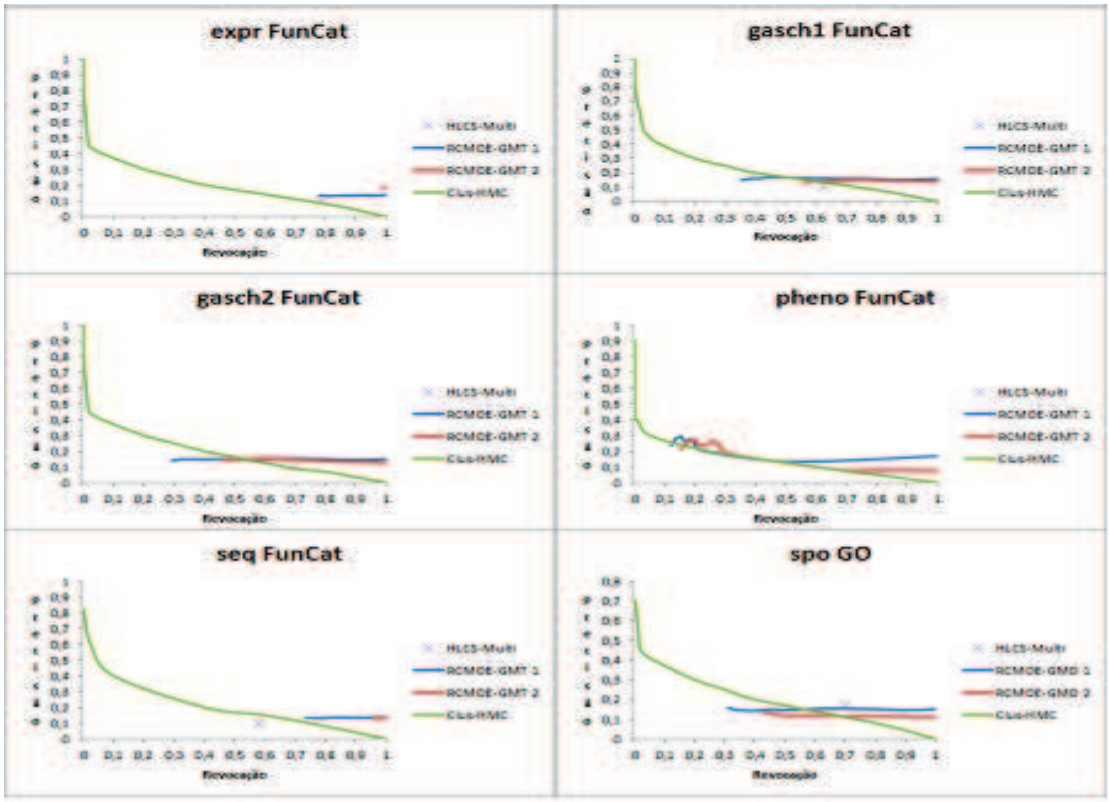

Figura 7.4 Curvas PR dos métodos RCM-GMT 1 e RCM-GMT 2 a partir de quatro das dez bases descritas na tabela 5.4

As curvas PR dos métodos RCM-GMT 1 e RCM-GMT 2 se apresentaram de forma semelhante à curva do Clus-HMC nos experimentos a partir da base pheno. Sob determinadas condições, nos experimentos a partir das bases church, derisi, eisen, gasch1, gasch2, pheno e spo, os métodos RCM-GMT 1 e RCM-GMT 2 e o método Clus- apresentaram medidas de precisão e revocação idênticas. Na tabela 7.15 são apresentadas as área sob cada curva PR ilustrada na figuras 7.3 e 7.4, com exceção das curvas (pontos) do método HLCS-Multi, já que não existe área sob um ponto. As áreas sob as curvas dos métodos RCM-GMT 1 e RCM-GMT 2, independente da base usada, foram inferiores à área sob as curvas do método Clus-HMC. A partir dos experimentos da base pheno, o valor da área sob a curva PR entre os três métodos foram próximos. Nos experimentos apresentados na tabela 7.15 o teste t de *student* não pôde ser usado para comparação entre os métodos RCM-GMT 1 e RCM-GMT 2, já que não foi construída uma curva para cada execução de cada base, e sim uma curva para cada base usando as médias entre as medidas de cada execução.

Tabela 7.19 Resultados dos experimentos dos métodos RCM-GMT e Clus-HMC.

Nome da base/Método	RCM-GMT 1	RCM-GMT 2	Clus-HMC
Cellcycle	0.095	0.058	0.172
Church	0.097	0.071	0.170
Derisi	0.100	0.065	0.175
Eisen	0.106	0.074	0.204
Expr	0.029	0.002	0.210
Gasch1	0.102	0.063	0.205
Gasch2	0.104	0.076	0.195
Pheno	0.147	0.122	0.160
Seq	0.035	0.006	0.211
Spo	0.102	0.066	0.186

Considerando que o valor máximo da área sob a curva PR máxima é 1[Vens et al., 2008], e 3% de 1 é igual a 0.03, o Clus-HMC superou o RCM-GMT 1 em todos os experimentos e superou o RCM-GMT 2 a partir de nove das dez bases (com exceção da Pheno). Além disso, o RCM-GMT 2 superou o RCM-GMT 1 a partir das bases Cellcycle, Derisi, Eisen, Gasch 1 e Spo. Usando o teste Kruskal Wallis, levando-se em consideração os experimentos de todas as bases juntas, o Clus-HMC apresentou resultados estatisticamente superiores aos métodos RCM-GMT 1 e RCM-GMT 2. Os resultados dos experimentos do principal método deste trabalho é descrito na seção 7.4.

7.4 Etapa 4

Os resultados dos experimentos do RCM-LMD e RCM-GMD a partir das dez bases usadas são apresentados na tabela 7.20. São marcados em negrito os resultados nos quais um método superou o outro. Independentemente da base usada, o RCM-GMD 1 superou o RCM-LMD com uso das medidas hR e hF. Usando a medida hF, o RCM-GMD1 superou o RCM-LMD a partir de sete das dez bases usadas. O RCM-

LMD só não foi superado pelo RCM-GMD 1 com o uso da medida hP a partir das bases church, derisi e pheno. Com uso das três medidas e as bases eisen e pheno, o RCM-LMD apresentou desvio padrão igual a zero, sendo que com o uso das outras bases o valor do desvio padrão foi igual a 0.1, independentemente da medida usada.

Tabela 7.20 Resultados dos experimentos dos métodos RCM-LMD e RCM-GMD.

	RCM-LMD			RCM-GMD 1		
Nome	hR	hP	hF	hR	hP	hF
Cellcycle	0.1367±0.01	0.3175±0.01	0.1911±0.01	**0.3387±0.01**	**0.3537±0.01**	**0.3459±0.01**
Church	0.1366±0.01	0.3169±0.01	0.1909±0.01	**0.3632±0.04**	0.3651±0.06	**0.3612±0.03**
Derisi	0.1358±0.01	0.3485±0.01	0.1955±0.01	**0.3387±0.01**	0.3559±0.03	**0.3465±0.01**
Eisen	0.1255±0.00	0.2080±0.00	0.1566±0.00	**0.3422±0.01**	**0.3528±0.02**	**0.3470±0.01**
Expr	0.1367±0.01	0.3191±0.01	0.1914±0.01	**0.3485±0.02**	**0.3505±0.02**	**0.3493±0.01**
Gasch1	0,1363±0.01	0,3178±0.01	0,1908±0.01	**0.3406±0.01**	**0.3440±0.02**	**0.3421±0.01**
Gasch	0,1370±0.01	0,3173±0.01	0,1913±0.01	**0.3463±0.01**	**0.3474±0.02**	**0.3467±0.01**
Pheno	0.1462±0.00	0.3564±0.00	0.2073±0.00	**0.3670±0.04**	0.3871±0.06	**0.3720±0.02**
Seq	0.1337±0.01	0.3136±0.01	0.1875±0.01	**0.3420±0.01**	**0.3511±0.02**	**0.3460±0.01**
Spo	0.1369±0.01	0.3182±0.01	0.1914±0.01	**0.3422±0.01**	**0.3532±0.02**	**0.3472±0.01**

Usando o teste de Mann-Whitney, conclui-se que o RCM-GMD 1 teve desempenho superior ao RCM-LMD nos resultados apresentados na tabela 7.20. Os resultados dos experimentos do RCM-GMD 1 e do HLCS-Multi a partir das dez bases usadas são apresentados na tabela 7.21. São marcados em negrito os resultados em que o RCM-GMD 1 superou o HLCS-Multi ou em que os dois obtiveram resultados semelhantes. Pode se observar que o RCM-GMD 1 superou ou obteve resultados semelhantes ao HLCS-Multi usando a medida hP a partir de todas as bases usadas. Além disso, usando a medida hR, o RCM-GMD superou o HLCS-Multi em uma das bases, sendo superado nas outras nove. Usando a medida hF, o RCM-GMD superou ou obteve resultados semelhantes ao HLCS-Multi em cinco das dez bases. Há diferença estatisticamente significante a favor do HLCS-Multi usando a medida hR e a favor do RCM-GMT usando a medida hP. Usando a medida hF e o teste de Mann-Whitney, concluiu-se que não houve diferença estatisticamente significante entre os resultados dos dois métodos.

Tabela 7.21 Resultados dos experimentos dos métodos RCM-GMD e HLCS-Multi.

	RCM-GMD 1			HLCS-Multi		
Nome	hR	hP	hF	hR	hP	hF
Cellcycle	0.3387±0.01	**0.3537±0.01**	0.3459±0.01	0.4209 ± 0.02	0.2611 ± 0.04	0.3223 ± 0.03
Church	0.3632±0.04	**0.3651±0.06**	0.3612±0.03	0.6183 ± 0.03	0.2259 ± 0.06	0.3309 ± 0.04
Derisi	0.3387±0.01	**0.3559±0.03**	0.3465±0.01	0.5655 ± 0.03	0.2948 ± 0.03	0.3873 ± 0.03
Eisen	**0.3422±0.01**	**0.3528±0.02**	0.3470±0.01	0.2407 ± 0.04	0.2206 ± 0.03	0.2302 ± 0.04
Expr	0.3485±0.02	**0.3505±0.02**	0.3493±0.01	0.3568 ± 0.03	0.3254 ± 0.05	0.3109 ± 0.04
Gasch1	0.3406±0.01	**0.3440±0.02**	0.3421±0.01	0.4203 ± 0.05	0.2577 ± 0.03	0.3195 ± 0.03
Gasch2	0.3463±0.01	**0.3474±0.02**	0.3467±0.01	0.4703 ± 0.03	0.2831 ± 0.02	0.3534 ± 0.04
Pheno	0.3670±0.04	**0.3871±0.06**	0.3720±0.02	0.7444 ± 0.04	0.2962 ± 0.04	0.4281 ± 0.05
Seq	0.3420±0.01	**0.3511±0.02**	0.3460±0.01	0.3547 ± 0.06	0.2347 ± 0.05	0.2825 ± 0.04
Spo	0.3422±0.01	**0.3532±0.02**	0.3472±0.01	0.3992 ± 0.04	0.2975 ± 0.03	0.3410 ± 0.04

Os resultados dos experimentos do RCM-GMD 1 e do RCM-GMD 2 a partir das dez bases usadas são apresentados na tabela 7.22 e 7.23. São marcados em negrito os resultados em que um dos métodos superou o outro.

Tabela 7.22 Resultados dos experimentos dos métodos RCM-GMD e HLCS-Multi.

	RCM-GMD 1			RCM-GMD 2		
Nome	hR	hP	hF	hR	hP	hF
Cellcycle	**0.3387±0.01**	**0.3537±0.01**	**0.3459±0.01**	**0.3387±0.01**	**0.3537±0.01**	**0.3459±0.01**
Church	0.3632±0.04	0.3651±0.06	0.3612±0.03	0.3632±0.04	0.3651±0.06	0.3612±0.03
Derisi	**0.3387±0.01**	**0.3559±0.03**	**0.3465±0.01**	**0.3387±0.01**	**0.3559±0.03**	**0.3465±0.01**
Eisen	**0.3422±0.01**	**0.3528±0.02**	**0.3470±0.01**	**0.3422±0.01**	**0.3528±0.02**	**0.3470±0.01**
Expr	0.3485±0.02	0.3505±0.02	0.3493±0.01	0.3485±0.02	0.3505±0.02	0.3493±0.01
Gasch1	**0.3406±0.01**	**0.3440±0.02**	**0.3421±0.01**	**0.3406±0.01**	**0.3440±0.02**	**0.3421±0.01**
Gasch2	0.3463±0.01	**0.3474±0.02**	0.3467±0.01	0.3463±0.01	**0.3474±0.02**	0.3467±0.01

Tabela 7.23 Resultados dos experimentos dos métodos RCM-GMD e HLCS-Multi.

	RCM-GMD 1			RCM-GMD 2		
Nome	hR	hP	hF	hR	hP	hF
Pheno	**0.3670±0.04**	**0.3871±0.06**	**0.3720±0.02**	**0.3670±0.04**	**0.3871±0.06**	**0.3720±0.02**
Seq	0.3420±0.01	0.3511±0.02	0.3460±0.01	0.3420±0.01	0.3511±0.02	0.3460±0.01
Spo	0.3422±0.01	0.3532±0.02	0.3472±0.01	0.3422±0.01	0.3532±0.02	0.3472±0.01

O RCM-GMD 2 apresentou resultados superiores aos do RCM-GMD 1 usando a base expr com o uso das medidas hR e hF. O RCM-GMD 1 superou o RCM-GMD 2 com o uso da três medida a partir das bases cellcycle, derisi, eisen, gasch1 e pheno e com o uso da medida hR a partir da base gasch2. Houve diferença estatisticamente significante a entre os resultados dos dois algoritmos usando as medidas hP e hF. Nos dois casos o RCM-GMD 1 apresentou resultados superiores. Na tabela 7.24 são apresentadas as médias de classes previstas pelos classificadores gerados pelos métodos RCM-LMD, RCM-GMD 1 e RCM-GMD 2.

Tabela 7.24 Médias do número de classes escolhidas pelos métodos RCM-LMD, RCM-GMD 1, RCM-GMD 2 e média do número de classes as quais as instâncias pertencem.

	RCM-LMD	RCM-GMD 1	RCM-GMD 2	Instância
Cellcycle	0,02	4,90	8,66	5,09
Church	0,01	5,05	8,47	5,08
Derisi	0,01	4,70	5,62	5,09
Eisen	0,00	5,11	6,22	5,26
Expr	0,10	4,88	11,95	5,08
Gasch1	0,04	4,73	5,89	5,08
Gasch2	0,01	4,94	8,45	5,08
Pheno	0,00	5,13	5,82	5,16
Seq	0,14	4,97	6,86	5,09
Spo	0,07	4,96	7,23	5,08
Média	0,04	4,94	7,52	5,11

Na quarta coluna são apresentadas as médias do número de classes as quais as instâncias pertencem. Para os métodos RCM-LMD eRCM-GMD 1 são levadas em consideração apenas as classes das regras que cobriram as instâncias. Assim, as

classes escolhidas pelo classificador quando o número de regras que cobrem a instância é menor do que o número de classes as quais a instância pertence são desconsideradas. Conforme descrito na tabela 7.24, nos experimentos feitos a partir de todas as tabelas o RCM-GMD 2 escolheu, em média, um número de classes maior do que o número de classes as quais cada instância pertence. Como pode ser observado, a partir de todas as bases o RCM-GMD 2 escolheu mais regras do que o RCM-GMD 1.

O número de regras geradas e a média de regras geradas por classe são apresentados nas tabelas 7.25 e 7.26. Em ambos os casos foram calculadas as médias entre os dez experimentos realizados. Usando os dados apresentados na tabela 7.20 pode se sugerir que o número de instâncias presentes na base tem influência no número de regras geradas pelo RCM-GMD, o que não acontece com o número de atributos da base. Isso pode ser observado principalmente por meio de três fatos: o primeiro deles é que as bases Derisi e Pheno tem números de atributos próximos e números de instâncias distantes, sendo que aquele com maior número de exemplos tem também o maior número de regras geradas. O segundo é que as bases Expr e Gasch1 têm número de instâncias e de regras geradas semelhantes e número de atributos distantes. O terceiro é que a base Church tem número de instâncias e de regras geradas maior do que a base Pheno, porém tem número de atributos menor.

Tabela 7.25 Número de classes geradas e média de classes geradas por classe a partir de todas as bases pelos métodos RCM-GMD 1 e RCM-GMD2.

Base	Número de regras	Média de regras por classe
Cellcycle	809,1±88,90	1,78±0,07
Church	205,7±11,00	1,39±0,08
Derisi	965,9±55,39	1,96±0,05
Eisen	605,4±47,95	1,68±0,04
Expr	865,9±67,93	1,81±0,06

Tabela 7.26 Número de classes geradas e média de classes geradas por classe a partir de todas as bases pelos métodos RCM-GMD 1 e RCM-GMD2.

Base	Número de regras	Média de regras por classe
Gasch1	1056,4±79,79	1,94±0,06
Gasch2	632,6±45,52	1,72±0,05
Pheno	51,2±5,85	1,10±0,06
Seq	849,3±45,36	1,80±0,05
Spo	600,8±25,13	1,68±0,04

Na figura 7.5 são apresentadas as curvas PR dos métodos RCM-GMD 1 e RCM-GMD 2 a partir de seis bases.

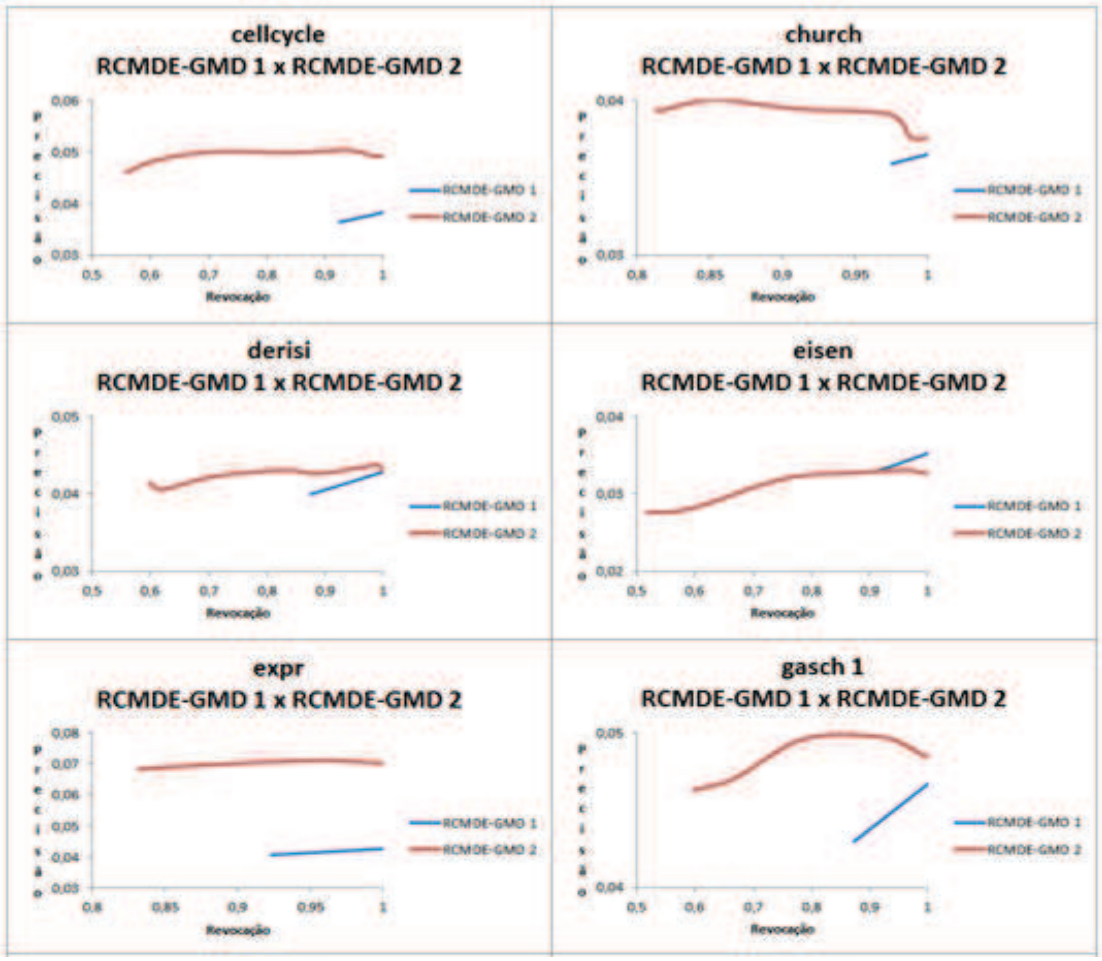

Figura 7.5 Curva PR dos experimentos dos métodos RCM-GMD 1 e RCM-GMD 2 a partir de seis das dez bases.

Na figura 7.6 são apresentadas as curvas PR dos métodos RCM-GMD 1 e RCM-GMD 2 a partir de quatro bases.

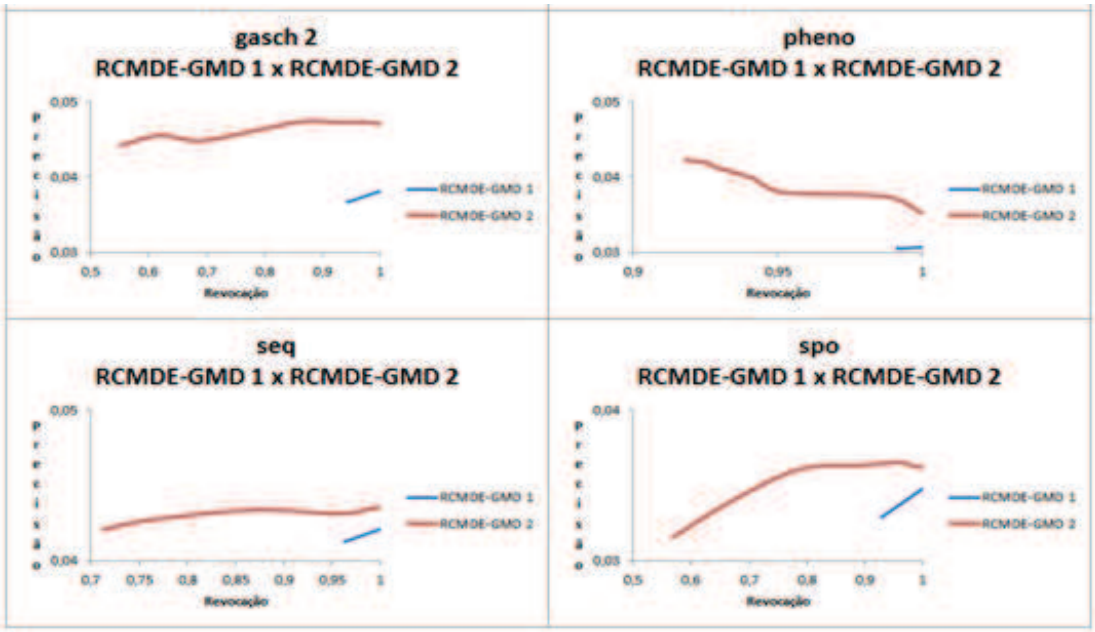

Figura 7.6 Curva PR dos experimentos dos métodos RCM-GMD 1 e RCM-GMD 2 a partir de 4 bases.

Com exceção da base Eisen, as curvas PR do método RCM-GMD 2 se apresentou acima das curvas do método RCM-GMD 1 (Tabelas 7.3 e 7.4).

Nas figuras 7.7 e 7.8 são apresentadas as curvas PR dos métodos HLCS-Multi, RCM-GMD 1, RCM-GMD 2 e Clus-HMC.

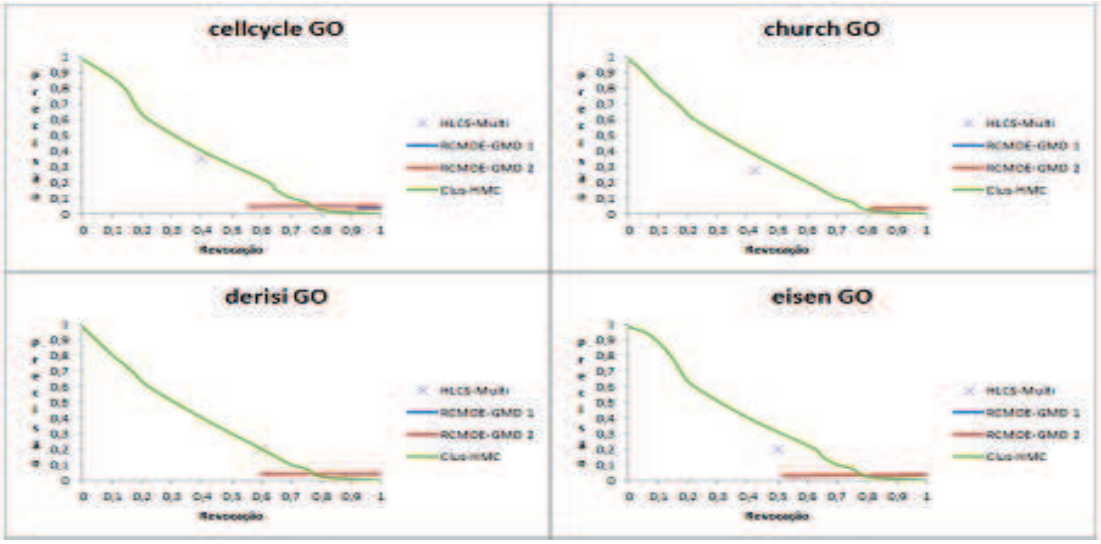

Figura 7.7 Curva PR dos experimentos dos métodos HLCS-Multi, RCM-GSD 1, RCM-GSD 2 e Clus-HMC a partir de 6 bases.

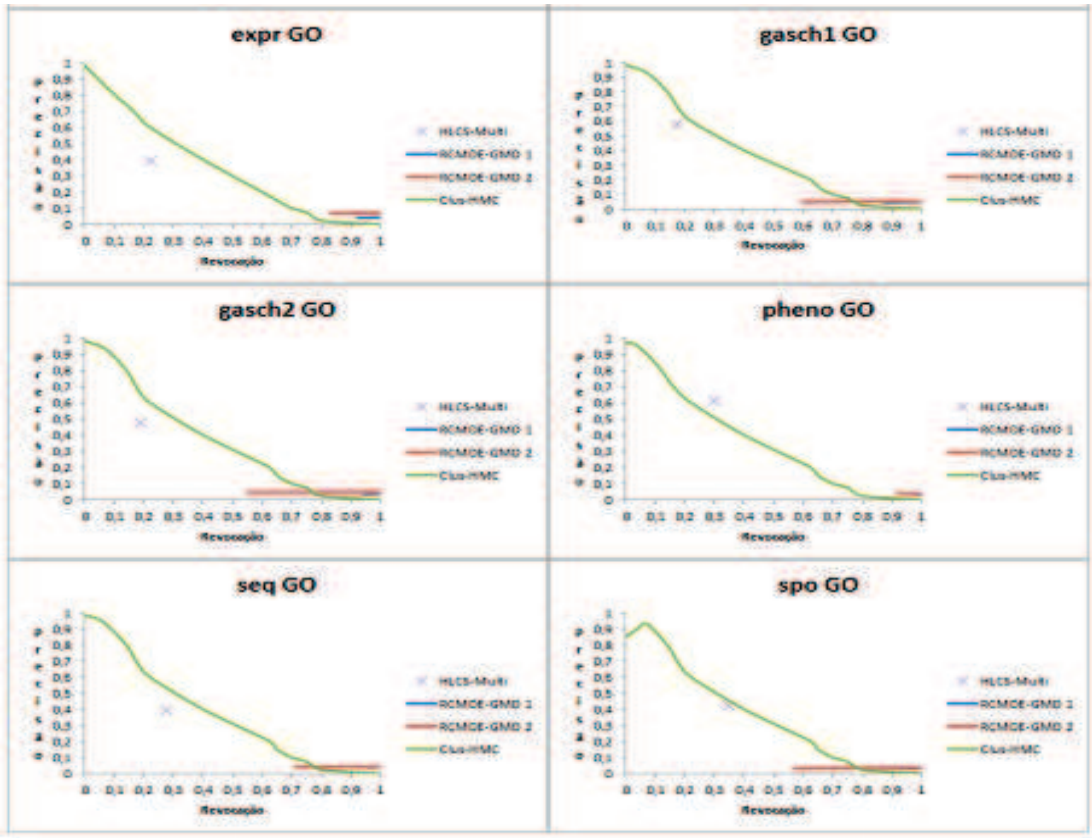

Figura 7.8 Curva PR dos experimentos dos métodos HLCS-Multi, RCM-GSD 1, RCM-GSD 2 e Clus-HMC a partir de 4 bases.

Independentemente da base e do limiar usados, os métodos RCM-GMD 1 e RCM-GMD 2 apresentaram a medida de revocação superior e precisão inferior ao HLCS-Multi. Além disso, independentemente da base usada, em alguma área do plano cartesiano, as curvas dos métodos RCM-GMD 1 e RCM-GMD 2 ficaram acima das curvas do Clus-HMC. Nesses casos os métodos RCM apresentaram resultados superiores.A área sob a curva PR dos experimentos feitos a partir de cada base com o uso do RCM-GMD 1 e do RCM-GMD 2 são apresentadas nas tabelas 7.27 e 7.28.

Tabela 7.27 Resultados dos experimentos dos métodos RCM-GMD e Clus-HMC.

Nome da base	RCM-GMD 1	RCM-GMD 2	Clus-HMC
Cellcycle	0.0029	0.0222	0.357
Church	0.0010	0.0074	0.348
Derisi	0.0051	0.0173	0.355

Tabela 7.28 Resultados dos experimentos dos métodos RCM-GMD e Clus-HMC.

Eisen	0.0033	0.0152	0.380
Expr	0.0033	0.0119	0.368
Gasch1	0.0058	0.0063	0.371
Gasch2	0.0022	0.0207	0.365
Pheno	0.0003	0.0035	0.337
Seq	0.0016	0.0124	0.386
Spo	0.0026	0.0154	0.352

O RCM-GMD 1 obteve resultados semelhantes ao RCM-GMD 2 a partir das bases gasch1. No uso de todas as outras bases o RCM-GMD 2 obteve melhores resultados do que o RCM-GMD 1. Como as curvas PR foram criadas a partir das médias entre os dez experimentos, somente uma curva por base foi criada e o teste t de *student* não pôde ser usado. Assim como nos experimentos dos métodos RCM-GMT 1 e RCM-GMT 2, um método foi considerado superior nos experimentos a partir de uma determinada base se o resultado foi, pelo menos, 0.03 acima dos resultados apresentados pelo outro algoritmo. Assim, o Clus-HMC superou estatisticamente os outros dois algoritmos a partir de todas as bases usadas e não houve diferença estatisticamente significante entre os experimentos dos métodos RCM-GMT 1 e RCM-GMT 2 a partir de nenhuma das bases.

7.5 Conclusões deste capítulo

7.5.1 RCM-LST X RCM-GST

No uso das bases GPCR, foi demonstrado com o uso dos resultados que não houve diferença entre os métodos em nenhum dos experimentos realizados. Porém, o método global se torna vantajoso devido à maneira como a classificação de um novo exemplo é feita, já que, para isso, enquanto o classificador local pode utilizar mais de um classificador, o classificador global utiliza apenas um, o que torna a tarefa mais simples e compreensível.

A partir das bases com exemplos de enzimas, o RCM-GST se mostrou vantajoso em relação ao RCM-LST também em relação ao desempenho. Sugere-se

que isso tenha ocorrido devido ao fato de o classificador global levar em consideração a hierarquia entre as classes de maneira natural, enquanto o método local separa os níveis da estrutura de classes e os trata sem interação entre si. Além de esse fato ser vantajoso por si só, ele possibilitou que as classes das regras tenham sido modificadas com o passar das gerações da maneira descrita na seção 3.2.1. O método global se mostrou mais sensível do que o local em relação ao número de exemplos da base de treinamento. Isso pode ser observado por meio dos dados apresentados na tabela 7.29.

Tabela 7.29 Compilação dos dados apresentados na tabela 7.1.

Grupo	Média do número de exemplos	Número de experimentos
GPCR	6539,25	12/0/0
Enzimas	14020	12/9/0

Na segunda coluna da tabela 7.21 são apresentadas as médias dos números de exemplos presentes na base de dados, agrupadas por tipo de proteína que compõe as bases. Assim, são apresentadas as médias entre as bases compostas por proteínas GPCR e entre as bases compostas por enzimas. A terceira coluna apresenta, separados pelo digito '/', o número de experimentos feitos usando as bases da linha correspondente, o número de experimentos nos quais o método global apresentou resultados estatisticamente superiores e o número de experimentos nos quais o método local foi estatisticamente superior. Foi considerado que a avaliação do algoritmo usando três medidas diferentes são três experimentos diferentes.

Pode se observar por meio dos dados apresentados na tabela 7.22 que quando o número de exemplos presentes na base de dados cresceu, o número de experimentos nos quais o método global foi estatisticamente superior também cresceu. Dessa forma, o comportamento do método global se mostrou mais influenciável e portanto mais sensível ao número de exemplos presentes na base do que o método local.

7.5.2 RCM-LST X LMNBwU

Apesar de não haver diferença estatisticamente significante entre os resultados, o RCM-LST apresentou resultados superiores ao LMNBwU nas bases com exemplos de enzimas. De acordo com os dados apresentados na tabela 7.30, o RCM-LST se mostrou mais sensível ao número de exemplos presentes na base do que o LMNBwU.

Tabela 7.30 Compilação doa dados apresentados na tabela 7.2.

Grupo	Média do número de exemplos	Número de experimentos
GPCR	6539,25	12/1/0
Enzimas	14020	12/12/0

A estrutura da tabela 7.22 é a mesma da tabela 7.21, sendo que, na terceira coluna estão, separados pelo dígito '/' o número de experimentos realizados usando os dois algoritmos, o número de experimentos nos quais o RCM-LST apresentou resultados estatisticamente superiores ao LMNBwU e o número de experimentos nos quais o LMNBwU apresentou resultados superiores ao LMNBwU. Por meio dos dados apresentados na tabela 7.22 se pode concluir que quando cresceu o número de exemplos presentes na base de dados o número de experimentos nos quais o RCM-LST superou o LMNBwU também cresceu. Dessa forma, o RCM-LST se mostrou mais sensível ao número de exemplos presentes na base de dados do que o LMNBwU. Assim, o RCM-GST apresentaram resultados satisfatórios quando comparado ao LMNBwU.

7.5.3 RCM-GST x GMNBwU x HLCS-Multi

Assim como nos experimentos contra o RCM-LST, nos experimentos contra o GMNBwU e o HLCS-Multi, o RCM-GST apresentou resultados mais satisfatórios com o uso das bases com exemplos de enzimas. Assim, o RCM-GST demonstrou, por meio dos resultados apresentados, que é mais sensível ao número de exemplos da base de treinamento do que os outros dois algoritmos. Sugere-se que o RCM-GST superou os outros dois algoritmos nas bases contendo enzimas, principalmente por dois motivos. O primeiro deles é a forma de avaliação de uma regra, que privilegia as regras que acertam mais e erram menos. O segundo é a estratégia de mutação descrita na seção 3.2.1, em que a nova classe do indivíduo se baseia nas classes dos exemplos em que ele cobre. O objetivo da nova classe do indivíduo é maximizar o número de casos em que as classes previstas e corretas são iguais.

7.5.4 RCM-LSD x RCM-GSD

Apesar de não ter havido diferença estatisticamente significante quando levadas em consideração todas as bases juntas, o RCM-GSD apresentou resultados superiores de maneira significante estatisticamente ao RCM-LSD em quatro das cinco

bases, usando pelo menos uma das medidas. Sugere-se que isso tenha ocorrido devido ao fato de o classificador global levar em consideração a hierarquia entre as classes de maneira natural.

7.5.5 RCM-GSD x hAnt-Miner x HLCS-DAG

Quando comparado aos algoritmos hAnt-Miner e HLCS-DAG, o RCM-GSD apresentou resultados satisfatórios, já que apenas no uso da base ds1 IntAct e a medida hP ele teve resultado inferior ao algoritmo usado para comparação. Nos outros casos, quando o RCM-GSD apresentou resultado inferior a um dos algoritmos, superou ou obteve resultados semelhantes ao outro. Nada foi concluído a respeito da sensibilidade do algoritmo ao número de exemplos presentes na base observada nas outras versões do RCM.

7.5.6 RCM-LMT x RCM-GMT 1

O RCM-GMT 1 demonstrou, por meio dos resultados apresentados, desempenho mais satisfatório do que o RCM-LMT. Sugere-se que isso tenha ocorrido devido à estratégia de mutação usada, feita como descrito na seção 4.6 e devido ao fato de a dependência entre as classes no método global ser levada em consideração de maneira natural, o que não ocorre no método local.

Por meio dos resultados apresentados na tabela 7.10 pode-se observar que os classificadores gerados pelo RCM-LMT preveem poucas classes (sem levar em consideração as classes escolhidas pelo classificador caso o número de classes previstas seja menor do que o número de classes corretas) quando comparados o número de classes previstas pelos classificadores gerados pelo RCM-GMT 1, RCM-GMT 2 e o número de classes corretas. Por meio disso, pode-se concluir que as classes previstas pelo RCM-LMT e consequentemente os acertos do classificador não foram obtidos com o uso das regras geradas, e sim por meio das classes mais presentes na base de dados, que são usadas caso as classes escolhidas pelo classificador seja menor do que o número de classes corretas.

7.5.7 RCM-GMT 1 x HLCS-Multi

Usando as medidas hR, hR e hF, o RCM-GMT 1 teve desempenho inferior ao HLCS-Multi. Observou-se que isso ocorreu porque, principalmente, nos experimentos realizados a partir da maioria das bases, as regras escolhidas pelo algoritmo cobrem

poucos exemplos e por isso têm pouca capacidade de generalização. Isso fez com que muitos erros tenham sido cometidos durante a fase de testes.

As regras mais bem avaliadas apresentaram baixa taxa de cobertura porque a maioria das regras geradas não tem ancestral em comum com nenhuma das classes de muitos exemplos. Assim, regras que cobrem três instâncias e que suas classes têm nós em comum com classes das três instâncias foram escolhidas para compor o classificador, já que demonstraram ter maior aptidão ao ambiente do que regras que, por exemplo, cobrem cem exemplos e possuem nós em comum com classes de noventa desses exemplos. Acredita-se que essa seja uma característica atípica da maioria das bases usadas nos experimentos.

Em relação às curvas PR, como o ponto apresentado no plano cartesiano representa as medidas de precisão e revocação do HLCS-Multi em todos os limiares usados, independentemente da base usada, o RCM-GMT 1, com o uso de determinados limiares, superou o HLCS-Multi quando a medida de revocação foi usada. A partir de algumas bases, com o uso de determinados limiares, o RCM-GMT 1 superou o HLCS-Multi no uso das medidas de revocação e de precisão.

7.5.8 RCM-GMT 1 x RCM-GMT 2

Nos experimentos realizados, usando as medidas hR, hP e hF, o RMCDE-GMT 1 superou, com diferença estatisticamente significante, o RCM-GMT 2. Sugere-se que isso tenha acontecido, principalmente, por dois motivos. O primeiro são as classes escolhidas pelo RCM-GMT 1 para compor o conjunto de classes previstas caso o número de classes previstas seja menor do que o número de classes as quais a instância pertence.

No momento da escolha das classes previstas por um classificador multirrótulo o qual as regras possuem apenas um consequente, três situações podem ser observadas. Na primeira delas, o número de regras que cobrem a instância é menor do que o número de classes as quais a instância pertence. Nesse caso, inicialmente, o número de classes previstas pelo RCM-GMT 1 é maior ou igual ao número de classes previstas pelo RCM-GMT 2, já que o RCM-GMT 2 usa um limiar para a escolha das classes previstas. Como, de acordo com a tabela 7.11, o RCM-GMT 2 previu, em média, mais classes do que o RCM-GMT 1, é sugerido que, nos casos em que o número de regras que cobrem a instância é menor do que o número de classes as quais

a instância pertence, o número de classes previstas pelos classificadores RCM-GMT 1 e RCM-GMT 2 seja igual. Nesse caso, as classes previstas foram as mesmas e nenhum dos dois classificadores levaria vantagem. Porém, o RCM-GMT 1 prevê novas classes previstas para que o número de classes previstas seja igual ao número de classes da instância. Sendo assim, o RCM-GMT 1, além de prever as classes previstas pelo RCM-GMT 2, prevê as classes mais presentes na base de dados. Dessa forma, o RCM-GMT 1 leva vantagem sobre o RCM-GMT 1.

Na segunda situação, o número de regras que cobrem a instância é igual ao número de classes as quais a instância pertence. Nesse caso, o RCM-GMT 1 escolhe as classes de todas as regras que cobrem a instância, enquanto o RCM-GMT 2 escolhe as regras que cobrem a instância e têm *fitness* maior ou igual a um determinado limiar. Nesse caso, o número de classes previstas pelo RCM-GMT 1 é maior ou igual ao número de classes previstas pelo RCM-GMT 2. Caso seja maior, as classes escolhidas pelo RCM-GMT 2 estão contidas no conjunto das classes escolhidas pelo RCM-GMT 1. Caso seja igual, as classes escolhidas pelo RCM-GMT 1 são as mesmas escolhidas pelo RCM-GMT 2. Como, de acordo com a tabela 7.11, o RCM-GMT 2 previu, em média, mais classes do que o RCM-GMT 1, é sugerido que, nos casos em que o número de regras que cobrem a instância é igual ao número de classes as quais a instância pertence, o número de classes previstas pelos classificadores RCM-GMT 1 e RCM-GMT 2 seja igual. Nesse caso, nenhum dos dois classificadores leva vantagem sobre o outro.

Na terceira situação, o número de regras que cobrem a instância é maior do que o número de classes as quais a instância pertence. Nesse caso, o RCM-GMT 1 escolhe um número de classes que seja igual ao número de classes as quais a instância pertence, levando em consideração o *fitness* das regras. O RCM-GMT 2 escolhe as regras com *fitness* maior ou igual a um determinado limiar. Assim, nesse caso, RCM-GMT 2 pode prever menos classes do que o RCM-GMT 1, mais classes do que o RCM-GMT 1 ou o mesmo número de classes do RCM-GMT 1. Como, de acordo com a tabela 7.11, o número médio de classes previstas pelo RCM-GMT 2 é maior do que o número médio de classes previstas pelo RCM-GMT 1, sugere-se que, nos casos em que o número de regras que cobriram a instância é maior do que o número de classes as quais a instância pertence, o número de classes previstas pelo RCM-GMT 2 é maior do que o número de classes previstas pelo O RCM-GMT 1. Nesse caso, as

classes previstas pelo O RCM-GMT 1 fazem parte do conjunto de classes previstas pelo O RCM-GMT 2. Sendo assim, o RCM-GMT 2 levou vantagem sobre o RCM-GMT 1.

Como, dentre as situações descritas, a única em que o RCM-GMT 1 levou vantagem sob o RCM-GMT 2 foi na primeira situação descrita, sugere-se que a escolha das novas classes previstas no caso em que o número de classes previstas pelo classificador é menor do que o número de classes da instância tenha sido o primeiro motivo pelo qual o RCM-GMT 1 superou o RCM-GMT 2 nos experimentos usando as medidas hR, hP, hF.

O segundo motivo sugerido que fez com que o RCM-GMT 1 tenha tido desempenho superior ao RCM-GMT 2 é a forma como as classes previstas e corretas são combinadas durante a fase de testes. Enquanto, no RCM-GMT 1 é usada a melhor combinação entre as classes previstas e corretas, no RCM-GMT 2, caso uma classe correta não seja prevista por nenhuma classe prevista, uma classe prevista é escolhida randomicamente e excluída do conjunto de classes previstas. Nesse caso, caso haja duas classes corretas e duas classes previstas, existe a possibilidade de a primeira classe correta ter a distância máxima em relação às duas classes previstas e a classe prevista a ser sorteada ser aquela em que a segunda classe correta é mais próxima.

Assim, como o RCM-GMT 1 e o RCM-GMT 2, em geral, previu um números de regras parecidos, o limiar usado no RCM-GMT 2 apresentou-se como uma alternativa para a escolha de classes previstas em classificadores hierárquicos multirrótulo.

7.5.9 RCM- GMT 1 x RCM- GMT 2 x Clus-HMC

Os métodos RCM-GMT 1 e RCM-GMT 2, independentemente da base usada, sob determinadas condições, apresentaram medidas de precisão e revocação superiores ao Clus-HMC, sendo que em alguns casos, os métodos RCM e o Clus-HMC apresentaram comportamentos parecidos. Como a curva PR não apresenta o limiar usado, os autores deste trabalho não executaram o Clus-HMC e toda a curva do RCM está acima de uma determinada faixa da curva do método Clus, pode se concluir que, sob o uso de determinados limiares, as medidas de precisão e a revocação dos métodos RCM-GMT 1 e RCM-GMT 2 foram superiores às mesmas medidas do Clus. Apesar de não terem apresentado desempenhos satisfatórios com o uso de medidas

hierárquicas, os métodos RCM-GMT 1 e RCM-GMT 2 demonstraram ter boa capacidade de predição de funções de proteínas como classificador binário.

7.5.10 RCM-LMD x RCM-GMD 1

O RCM-GMD 1 demonstrou, por meio dos resultados apresentados, desempenho mais satisfatório do que o RCM-LMD. Sugere-se que isso tenha ocorrido devido ao fato de a dependência entre as classes no método global ser levada em consideração de maneira natural, o que não ocorre no método local. Por meio dos resultados apresentados na tabela 7.12 pode-se observar que as classes previstas pelo RCM-LMD e consequentemente os acertos do classificador não foram obtidos com o uso das regras geradas, e sim por meio das classes mais presentes na base de dados, que são usadas caso as classes escolhidas pelo classificador seja menor do que o número de classes corretas. Isso ocorreu pelos mesmos motivos que o fato ocorreu para o RCM-LMT, descritos na seção 7.1.5. Pode se observar que no uso das bases pheno e eisen o RCM-LMD não conseguiu prever nenhuma classe durante toda a fase de teste. Esse comportamento do método é refletido no desvio padrão dos resultados mostrados na tabela 7.15, já que, como todas as classes previstas pelo método foram as mais instanciadas na base de treino e a base foi a mesma nas 10 execuções, não houve variação nos resultados.

7.5.11 RCM- GMD 1 x HLCS-Multi

O RCM-GMD 1 apresentou resultados satisfatórios quando comparado ao HLCS-Multi. Apesar de não ter tido diferença estatisticamente entre os métodos no uso da medida hF, o RCM-GMD 1 superou o HLCS-Multi de maneira estatisticamente significante no uso da medida hP, com resultados superiores a partir de todas as bases. O HLCS-Multi superou o RCM-GMD 1 de maneira estatisticamente significante no uso da medida hR, com resultados superiores a partir de todas as bases. Isso ocorreu devido ao fato de o HLCS-Multi ter apresentado resultados com a medida hR significativamente superiores aos resultados com a medida hP, já que os resultados apresentados pelo RCM-GMD usando as duas medidas foram semelhantes.

Com o uso da revocação não hierárquica, o RCM-GMD 1 apresentou resultados superiores ao HLCS-Multi. Isso ocorreu por conta do baixo número de

Falsos Negativos, já que os indivíduos apresentaram *fitness* altos e, consequentemente, em poucos casos o classificador previu uma instância como pertencente à classe "Negativo". Com o uso da precisão não hierárquica, o RCM-GMD 2 apresentou resultados inferiores ao HLCS-Multi. Como o classificador previu muitas instâncias como pertencente à classe "Positivo", por conta do alto fitness das regras geradas, houve um alto número de Falsos Positivos, diminuindo a precisão não hierárquica.

7.5.12 RCM- GMD 1 x RCM- GMD 2

Nos experimentos realizados, usando as medidas hR, hP, hF e Ac, o RMCDE-GMD 1 superou, com diferença estatisticamente significante, o RCM-GMD 2. Sugere-se que isso tenha acontecido, principalmente, por dois motivos, sendo eles os mesmos descritos na seção 7.1.7 para os casos dos métodos RCM-GMT 1 e RCM-GMT 2. Assim como o limiar usado no RCM-GMT 2, o limiar usado no RCM-GMD 2 apresentou-se como uma alternativa para a escolha de classes previstas em classificadores hierárquicos multirrótulo.

7.5.13 RCM- GMD 1 x RCM- GMD 2 x Clus-HMC

Sob determinadas condições, os métodos RCM- GMD 1 x RCM- GMD 2 apresentaram medidas de precisão e revocação não hierárquicas superiores ao Clus-HMC. Porém, em termos gerais, a medida de precisão não hierárquica dos métodos RCM- GMD 1 x RCM- GMD 2 foi inferior à mesma medida no Clus-HMC. Conforme mencionado na seção anterior, isso ocorreu devido ao número elevado Falsos Positivos. Pelos mesmos motivos apresentados para o RCM-GMT 1 e RCM-GMT 2, pode se concluir que, sob o uso de determinados limiares, as medidas de precisão e a revocação dos métodos RCM-GMD 1 e RCM-GMD 2 foram superiores às mesmas medidas do Clus-HMC.

Na tabela 7.23 é apresentada uma comparação entre todos os métodos usados nos experimentos deste trabalho. Os métodos LMNBwU e GMNBwU são referenciados como xMNBwU. Em cada célula da tabela, caso se aplique, estão, nessa ordem e separados por '/', o número de experimentos feitos para comparação entre o RCM e o método da linha correspondente, o número de experimentos em que o método em questão foi estatisticamente superior ao RCM e o número de experimentos

em o RCM foi superior ao método em questão. A avaliação dos métodos usando três medidas diferentes foi considerada como sendo três experimentos diferentes. Para a comparação com o Clus foi levada em consideração a área sob a curva PR.

Tabela 7.31 Tabela comparativa entre os métodos RCM e os outros métodos usados nos experimentos deste trabalho.

Método/Problema	LST	GST	LSD	GSD	LMT	GMT	LMD	GMD
xMNBwU	3/0/0	3/0/0	-	-	-	-	-	-
HLCS	-	3/0/0	-	3/0/0	-	3/1/1	-	3/1/1
Clus	-	-	-	-	-	1/0/1	-	1/0/1
hAntMiner	-	-	-	3/0/0	-	-	-	-

Como pode ser observado na tabela 7.23, em dois casos o RCM superou estatisticamente o método usado para comparação e em 4 casos o método usado superou o RCM. Em todos os outros casos não houve diferença estatisticamente significante. Portanto, em todas as execuções o RCM se mostrou competitivo em relação ao outro método já que, em 3 delas nem o RCM nem o algoritmo usado se mostrou vantajoso (LST, GST e GSD) e nas outras duas (GMTxHLCS e GMDxHLCS) em um dos casos o RCM foi estatisticamente superior, em outro caso o HLCS foi superior e em um terceiro caso não houve diferença estatisticamente significativa entre os métodos. Apesar de o Clus ter levado vantagem estatisticamente significativa quando a área sob a curva PR foi usada, no uso da curva PR o RCM foi superior quando alguns limiares foram usados.

A tabela 7.32 apresenta para qual(is) tipo(s) de problema(s) o RCM e os métodos usados como comparação constroem classificadores. Portanto, as linha da tabela 7.32, a partir da segunda, correspondem aos métodos RCM e aqueles usados neste trabalho para comparação com o RCM e as colunas, a partir da segunda, correspondem aos possíveis tipos de problemas a serem apresentados. Caso o método apresentado preveja a resolução de um determinado tipo de problema, a célula que corresponde ao método e ao tipo de problema tem como conteúdo o dígito ‘X’. Caso contrário, a célula está vazia.

Tabela 7.32 Tabela comparativa entre os métodos RCM e os outros métodos usados nos experimentos deste trabalho.

Método/Problema	LST	GST	LSD	GSD	LMT	GMT	LMD	GMD
RCM	X	X	X	X	X	X	X	X
xMNB	X	X						
HLCS		X	X	X		X		X
Clus		X		X		X		X
Hantminer				X		X		X

Como pode ser visto na tabela 7.32, o RCM é o único método dentre os apresentados que constrói classificadores para todos os tipos de problemas apresentados.

7.6 Conclusões deste capítulo

Neste capítulo foram descritos e comentados os resultados dos experimentos usando todas as versões do RCM. Além disso, foram apresentadas duas tabelas (7.31 e 7.32) com compilações dos resultados. No capítulo 8 são descritas as conclusões deste trabalho com o objetivo de, com o auxílio dos resultados apresentados neste capítulo, verificar se a hipótese é verdadeira e se os objetivos foram atingidos.

Capítulo 8

Conclusões e trabalhos futuros

A partir dos resultados apresentados se pode concluir que a hipótese apresentada é verdadeira. Assim, a Evolução Diferencial pode ser usada em problemas com valores discretos, mais especificamente na construção de classificadores hierárquicos das funções de proteínas. O objetivo principal deste trabalho é o desenvolvimento de um método construtor de classificadores de proteínas de maneira independente das características apresentadas pelo problema. Portanto, já que, de acordo com os resultados apresentados, o objetivo foi cumprido, o método, chamado RCM, constrói classificadores locais e globais, de rótulo simples e multirrótulo e para uma hierarquia entre as classes representada por uma árvore ou por um DAG. Independentemente das características apresentadas o RCM se mostrou como uma alternativa aos métodos já existentes na literatura. O RCM é o primeiro método a usar a ED para a construção de classificadores hierárquicos e o primeiro a ser independente das características da base de dados apresentada.

Como pode ser observado, o RCM é o único método dentre os apresentados que prevê todos os tipos de problema. Não foi encontrado outro método na literatura que preveja todos os tipos de problema. Apesar do número de regras que compõem os classificadores gerados pelos métodos RCM ser alto, deve ser considerado o número de regras escolhidas para cada classificação, já que o RCM foi desenvolvido para ser usado por biólogos no auxílio à tomada de decisão durante o processo de classificação das proteínas. Assim, ao auxiliar o processo de classificação de uma nova proteína, a(s) regra(s) usada(s) pelo classificador para aquela proteína deve ser apresentada ao biólogo, juntamente com a(s) classe(s) a(s) qual(s) a proteína pertence.

Conclui-se portanto que o RCM é um método que pode ser usado para a construção de classificadores hierárquicos da função de proteínas de maneira não dependente das características da base de dados que armazena as instâncias e que consegue expressar o conhecimento usado para a classificação de maneira clara e expressiva. Além disso, o RCM apresentou resultados competitivos em relação aos

outros métodos usados para comparação. Nenhum outro método disponível atualmente na literatura possui as três características citadas.

Como trabalhos futuros, sugere-se o desenvolvimento de uma estratégia de mutação do consequente das regras representadas pelos indivíduos dos métodos globais com a hierarquia de classes representada por um DAG semelhante à feita pelos métodos globais com a hierarquia de classes representada por uma árvore. Além disso, as funções de fitness dos indivíduos podem ser estudadas de maneira mais aprofundada com o objetivo de aumentar o desempenho dos métodos. Sugere-se também o uso de novas estratégias de mutação dos antecedentes das regras representadas pelos indivíduos.

Referências

[Alberts et al., 2007] Alberts,B., Johnson,A., Lewis, J., Raff, M., Roberts, K., Walter, P. (2007), *Mollecular Biology of the cell*, 5th Edition.

[Ashburner et al., 2000] Ashburner, M.; Ball, C. a.; Blake, J. a.; Botstein, D., Butler, H.; Cherry,J. M.; Davis, a. P.; Dolinski, K.; Dwight, S. S.; Eppig, J. T.; Harris, M. a.; Hill, D. P.; Issel-Tarver, L.; Kasarskis, a.; Lewis S.; Matese, J. C.; Richardson, J. E.; Ringwald, M.; Rubin,G. M.; Sherlock, G.(2000), Gene ontology: tool for the unification of biology. The Gene Ontology Consortium, in:*Nature genetics*, volume 25, number 1, pages 25–29.

[Bi and Kwok, 2010] Bi, W. and Kwok, J.T.(2010), Multi-label classification on tree- and DAG structured hierarchy, in: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 17-24.

[Blockeel et al., 2002] Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., Struyf, J. (2002), Hierarchical multi;classification, in: Prooceeding of ACM SIGKDD 2002 workshop on multi-relational data mining (MRDM 2002), pages 21-35.

[Blockeel et al., 2006] Blockeel, H., Schietgat, L., Struyf, J., Dzeroki, S., Clare, A. (2006),*Decision Trees for Hierarchical Multi-label Classification: A Case Study in Functional Genomics*, Springer Berlin Heidelberg, pages 18-29.

[Borges, 2012] Borges, H.B. (2012)*, Classificador hierárquico multirrótulo usando uma rede neural competitiva, (Tese de doutorado, Pontifícia Universidade Católica do Paraná)*.

[Borges et al., 2013] Borges, H.B., Sillla Jr., C.N., Nievola, J.C. (2013), An evaluation of global-model hierarchical classification algorithms for hierarchical classification problems with single paths of labels, in: *Computers & Mathematics with Applications*, volume 66, number 10, pages 1991-2002.

[Brest et al., 2006] J. Brest, S. Greiner, B. Boškovi´c, M. Mernik, and V. Žumer (2006), Self-adapting control parameters in differential evolution: A comparative studyon numerical benchmark problems, in: *IEEE Transactions on EvolutionaryComputation*, volume 10, number 6, pages 646–657.

[Brest and Maučec, 2008] Brest, J., Maučec, M.S. (2008), Population size reduction for the differential evolution algorithm (2008), in: *Applied Intelligence*, volume 29, number 3, pages 228-247.

[Buhry et al., 2009] Buhry, L. and Giremus, A. and Grivel,E. Saıghi,S. and Renaud, S (2009), New variants of the Differential Evolution algorithm: application for neuroscientists, in: *Proceedings of EUSIPCO*, pages 2352-2356

[Cerri and Carvalho, 2011]Cerri, R. and de Carvalho,A.C.P.L.F. (2011),Hierarchical multi-label protein function prediction using local neural networks,in: *Advances in Bioinformatics and Computational Biology*, Springer Berlin Heidelberg, pages 10-17.

[Cerri et al., 2012] Cerri, R. Barros,R.C. and de Carvalho,A.C.P.L.F. (2012) A Genetic Algorithm for hierarchical multi-label classification, in: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 250-255.

[Clare and King, 2003] Clare, A., King, R.D., Predicting Gene Function in Saccharomyces Cerevisiae, in: Proceedings of the European Conference on Computational Biology, Vol. 19, Suppl 2, pages ii42-ii49.

[Coelho, 2008] Coelho, L.S.(2008) Reliability–redundancy optimization by means of a chaotic differential evolution approach, in: *Chaos, Solitons and Fractals*, volume 41, number 2, pages 594–602.

[Costa et al., 2007] E.P. Costa, A.C. Lorena, A.C.P.L.F. Carvalho, and A.A. Freitas (2007). A review of performance evaluation measures for hierarchical classifiers, in: *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop*, AAAI Technical Report WS-07-05, AAAI Press, pages 1-6.

[Das et al., 2009] Das,S. and Abraham, A. and Chakraborty, U. K. and Konar, A. (2009), Differential Evolution using a neighborhood-based mutation operator, in: *IEEE Transactions on Evolutionary Computation*, volume 13, number 3, pages 526, 553.

[Davies et al., 2007] Davies, M.N., Secker, A., Freitas, A.A., Mendao, M., Timmis, J. and Flower, D.R. (2007), On the hierarchical classification of G-Protein-Coupled Receptors, in:*Bioinformatics*, volume 23, number 23, pages 3113–3118.

[Devlin, 2011] Devlin,T.M., (2011), *Textbook of Biochemical with Clinical Correlations*, John Wiley & sons, Inc.

[Dumains and Chen, 2000] Dumais, S.T., Chen, H. (2000), Hierarchical classification of Web content. In:*Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256-263.

[Enright et al., 1999] Enright,A.J., Iliopoulos,I., Kyrpides,N.C. and Ouzounis,C.A. (1999),Protein interaction maps for complete genomes based on gene fusion events. In: *Nature*, volume 402 number 6757, pages 86–90.

[Epitropakis et al., 2010] Epitropakis, M.G. Tasoulis, D.K. Pavlidis, N.G.Plagianakos,V.P. Vrahatis, M.N.(2010),Enhancing Differential Evolution utilizingproximity-based mutation operators, in: *Evolutionary Computation*, volume 15, number 1, pages 99-119.

[Ettema et al., 2001] Ettema,T., Oost,J. and Huynen,M. (2001), Modularity in the gain and loss of genes: application for function prediction, in:Trends in Genetics., volume17, number 9, pages 485–487.

[Feoktistov, 2006] Feoktistov, V. (2006) Differential Evolution: In Search Solutions, Optimization And Its Applications, Springer, vol. 5.

[Freitas and Carvalho, 2007] Freitas, A.A., de Carvalho, A.C.P.L.F. (2007), A tutorial on hierarchical classification with applications in bioinformatics, in: Research and Trends in Data Mining Technologies and Applications, Edited by: Taniar D. Idea Group, pages 175-208.

[Gämperle et al., 2002] Gämperle, R., Müller,S.D and Koumoutsakos, P. (2002), A parameter study for differential evolution, in: *Advances in intelligent systems, fuzzy systems, evolutionary computation*, volume 10, pages 293-298.

[Guimarães, 2008] Guimarães, P.R.B. (2008), *Métodos Quantitativos Descritivos*, IESDE Brasil, Curitiba, Brasil.

[Gruber , 1993] Gruber, T.R, A Translation Approach to Portable Ontology Specifications, in: *Knowledge Acquisition*, volume 5, number 2, pages 199-220.

[Gruber , 2009] Gruber, T.R, Ontology, in: *The encyclopedia of database systems*, Liu, L and Oszu, M.T., Srpinger Verlag.

[Hand et al., 2001] Hand, D.J., Mannila, H. and Smyth, P. (2001), *Principles of Data Mining*, The MIT Press.

[Holden and Freitas, 2008a] Holden N., Freitas A.A., (2008a), Hierarchical classification of protein function with ensembles of rules and Particle SwarmOptimization, in:*Journal Soft Computing - A Fusion of Foundations, Methodologies and Applications - Special Issue on Evolutionary and Metaheuristics based Data Mining (EMBDM)*, volume 13, number 3, pages 259-272.

[Holden and Freitas, 2008b] Holden N., and Freitas A.A. (2008b), Hierarchical classification of G-Protein-Coupled-Receptors with a PSO/ACO algorithm, in: *Proceedings of the 2006 IEEE Swarm Intelligence Symposium*, pages 77-84.

[Holden and Freitas, 2009] Holden N., and Freitas A.A. (2009), Hierarchical classification of protein function with ensembles of rules and particle swarm optimisation, in: *Soft Computing, volume 13, number 3,* pages 259-272.

[Islam et al., 2012] Islam, S.M., Das, S., Ghosh, S., Roy, S. and Suganthan, P. N. (2012). An Adaptive Differential Evolution Algorithm with Novel Mutation and Crossover Strategies for Global Numerical Optimization, in:*Systems, Manufacturing and Cybernetics*, volume 42, number 2, pages 482-500.

[Kiritchenko et al., 2005] Kiritchenko, S., Matwin S., and Famili, F. (2005), Functional annotation of genes using hierarchical text categorization, in: *Proceedings of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (held at ISBM-05), Detroit, USA, 2005*.

[Kreutz et al., 2010] Kreutz,J.E.,Shukhaev,A.,Du, W., Druskin, S.,Daugulis,O., andIsmagilov,R.,F.(2010), Evolution of catalysts directed by Genetic Algorithms in aplug-based microfluidic device tested with oxidation of methane by oxygen, in:*Journal of the American Chemical Society*, volume 132, number 9, pages 3128-3132.

[Liu et al., 1999] Liu, H, Cai, Z. Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. (2010), in: *Applied Soft Computing* , volume 10, number 2, pages 229–240.

[Marcotte, et al., 1999] Marcotte,E.M., Pellegrini,M., Ng,H.L., Rice,D.W., Yeates,T.O. and Eisenberg,D. (1999), Detecting protein function and protein-protein interaction from genome sequences,in:*Science*, number 285, pages 751–753.

[Maulik, 2009] Maulik, U.(2009), Medical image segmentation using Genetic Algorithms, in:*Information Technology in Biomedicine*, volume 13, number2, pages 166-173.

[Metz, et al., 2011] Metz., J, Freitas, A.A., Monard, M.C., Cherman, E.A. (2011), A study on the selection of local training sets for hierarchical classification tasks, in*: Brazilian National Meeting On Artificial Intelligence (ENIA 2011)*, Sociedade Brasileira de Computação-SBC, pages 572-583.

[Miller, 1956] Miller, G.A., 1956. The Magical Number Seven, Plus or Minus Two Some Limit on Our Capacity for Processing Information, Psychological Review, volume 101 number 2, pages 343-352.

[Nelson and Cox, 2002]Nelson,D.,L., Cox, M.,M.(2002), *Lehninger Princípios de Bioquímica*, Terceira Edição.

[Noman, et al., 2011] Noman, N.,Bollegala, D.,Iba, H.(2011), Differential Evolution with Self Adaptive Local Search, in: Proceedings of the 13th annual conference on Geneticand evolutionary computation, pages 1099-1106.

[Olson and Delen, 2008]Olson, D. L.. Delen, D. (2008) *Advanced Data Mining Techniques*. Primeira edição, Springer.

[Otero et al, 2009] Otero, F.E.B., Freitas, A.A., Johnson, C.G., A hierarchical classification ant colony algorithm for predicting gene ontology terms*, in:Proceeding of the 7th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBio 2009), Lecture Notes in Computer Science 5483,* pages 339-357, Springer.

[Otero et al, 2010] Otero, F.E.B., Freitas, A.A., Johnson, C.G., A hierarchical multi-label classification*Ant Colony Algorithm for protein function prediction, in:Memetic Computing, volume 2, number 3, pages 165-181.*

[Overbeek et al., 1999] Overbeek,R., Fonstein,M., Souza,M.D., Pusch,G.D. and Maltsev,N. (1999), The use of gene clusters to infer functional coupling, in:*Proceedingsof National Academy of Sciences,* volume96, number 6, pages 2896–2901.

[Panduro et al.,2009] Panduro, M.A., Brizuela, C.A., Balderas, L.I., Acosta,D.A.(2009), A comparison of Genetic Algorithms, Particle Swarm Optmization and theDifferential Evolution Method for the design of scannable circular antenna arrays,Progress, in: Electromagnetics Research B, volume 13, pages 171-186.

[Qin et al. 2009] Qin , A. K., Huang V. L., andSuganthan P. N. (2009), Differential evolutionalgorithm with strategy adaptation for global numerical optimization, in: IEEE Transactions ofEvolutionary Computation, volume 13, number 2, pages 398–417.

[Romão, 2012] Romão, L.M. (2012), *Classificação Global Hierárquica Multirrótulo da função de proteínas utilizando Sistemas Classificadores, (Tese de doutorado, Pontificia Universidade Católica do Paraná).*

[Ruepp et al., 2004]Ruepp A, Zollner A, Maier D, Albermann K, Hani J, Mokrejs M, Tetko I, Guldener U, Mannhaupt G, Munsterkotter M, Mewes H. (2004), The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes, in:*Nucleic Acid Research*, volume 32, number 18, pages 5539–5545.

[Salama and Freitas, 2013]Salama, K.M and Freitas, A.A., (2013). ACO-Based bayesian network ensembles for the hierarchical classification of ageing-related proteins, in:*Proceedings of the 11th European conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 80-91.

[Salomon et al., 2000] Salomon, Perrin, G.R., Heitz, F. (2000). Differential Evolution for medical image registration, in: *International Conference of Artificial Intelligence*, pages 201-207.

[Santos et al., 1999] Santos, R.T., Nievola, J.C., Freitas, A.A., Lopes, H.S.,(1999), Extração de regras de Redes Neurais usando Algoritmos Genéticos, in: *Procedings of the IV BazilianConferenceon Neural Networks*, pages 158,163.

[Secker et al., 2010] A. Secker, M.N. Davies, A.A. Freitas, E. Clark, J. Timmis, and D.R. Flower (2010), Hierarchical classification of G-Protein-Coupled Receptors with data-driven selection of attributes and classifiers, in:*International Journal of Data Mining and Bioinformatics*, volume 4, pages 191-200.

[Silla and Freitas, 2009] Silla Jr., C.N., Freitas,A.A. (2009), A Global-Model Naive Bayes approach to the hierarchical prediction of protein functions,in:*Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, pages 992-997.

[Silla and Freitas, 2011] Silla Jr., C.N.S,A. Freitas, A.(2011), A survey of hierarchical classification across different application domains, in: *Data Mining and Knowledge Discovery*, volume 22, number 1-2, pages 31-71.

[Storn and Price, 1995] Storn, R., Price, K. (1995), *Differential Evolution - A simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*, Berkeley, ICSI.

[Storn, 1996] Storn, R. (1996), On the usage of Differential Evolution for functionoptimization, in:*Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American*, pages 519,523.

[Storn et al., 2005] Storn,R., Price,K. V., and Lampinen,J. (2005), *Differential Evolution–A Practical Approach to Global Optimization*, Springer.

[Theodorsson-Norheim, 1986] Theodorsson-Norheim, E. Kruskall-Wallis test: BASIC computer program to perform nonparametric one-way analysis variance and multiple comparisons on ranks of several independents samples, in: *Computer methods and programs in biomedicine*, volume 12, number 1, pages 57-62.

[Veterstrom and Thomsen, 2004] Veterstrom, J., Thomsen, T. (2004), A comparative study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on numerical benchmarks problems, in: *Congress on Evolutionary Computation*, volume 2, pages 1980-1987.

[Vens et al., 2008] Vens, C., Struyf, J., Shietgat, L.Dzeroski, S., Blockeel, H. (2008), Decision trees for hierarchical multi-label classification, in: *Machine Learning*, volume 73, number 2, pages 185-214.

[Zaharie, D. 2007] Zaharie, D. (2007), A comparative analysis of crossover variants in Differential Evolution, in: *Proceedings of International Multiconference on Computer Science and Information Technology*, pages 171-181.

[Zamuda and Brest, 2012]Zamuda, A. and Brest, J. (2012), Population reduction Differential Evolution with multiple mutation strategies in real world industry Challenges, in: *Swarm and Evolutionary Computation,* pages 154–161.

[Zamuda et al., 2013] Zamuda, A., Brest, J. and Mezura-Montes, E.(2013), Structured population size reduction Differential Evolution with multiple mutation strategies on CEC Ream Parameter Optimization, in: *Congress on Evolutionary Computation*, pages 1925-1931.

[Zhang and Sanderson, 2009] Zhang, J. and Sanderson, A.C. (2009), JADE: Adaptive differential evolution with optional external archive, in: *IEEE Transactions on Evolutionary Computation*, volume 13, number 5, pages 945–958.

[Zheng et al., 2002] Zheng,Y., Roberts,R.J. and Kasif,S. (2002), Genomic functional annotation using co-evolution profiles of gene clusters, in:*Genome Biol*., volume 3, number 11, pages 1-0060.