UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
PROGRAMA DE MESTRADO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

PROGRAMAÇÃO GENÉTICA APLICADA NO PROCESSO DE DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS DE REDES DE PESQUISA

Kedma Batista Duarte

Orientador: Prof. Dr. Leonardo da Cunha Brito

Goiânia
2010

KEDMA BATISTA DUARTE

PROGRAMAÇÃO GENÉTICA APLICADA NO PROCESSO DE DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS DE REDES DE PESQUISA

Dissertação apresentada ao Programa de Mestrado em Engenharia Elétrica e de Computação, da Escola de Engenharia Elétrica e de Computação, da Universidade Federal de Goiás, para obtenção do título de mestre em Engenharia Elétrica e de Computação.

Área de Concentração: Engenharia de Computação

Orientador: Prof. Dr. Leonardo da Cunha Brito

Goiânia
2010

Dados Internacionais de Catalogação na Publicação na (CIP)
GPT/BC/UFG

Duarte, Kedma Batista.
D812p Programação genética aplicada no processo de descoberta de conhecimento em bases de dados de redes de pesquisa [manuscrito] / Kedma Batista Duarte. - 2010.
xviii, 86 f. : il., figs, tabs.

Orientador: Prof. Dr. Leonardo da Cunha Brito.
Dissertação (Mestrado) – Universidade Federal de Goiás, Escola de Engenharia Elétrica e de Computação, 2010.
Bibliografia: f.84-86.
Inclui lista de figuras, tabelas, quadros e abreviaturas.
Anexos.

1. Programação genética (Computação). 2. Mineração de dados (Computação). 3. Evolução (Biologia) – Pesquisa I. Título.

CDU: 004:575.827

Universidade Federal de Goiás
Escola de Engenharia Elétrica e de Computação
Coordenação do Programa de Pós-Graduação em Engenharia Elétrica e de Computação

FOLHA DE APROVAÇÃO

"Programação Genética Aplicada no Processo de Descoberta de Conhecimento em Bases de Dados de Redes de Pesquisa"

KEDMA BATISTA DUARTE

Dissertação defendida e aprovada pela banca examinadora constituída pelos senhores:

Leonardo da Cunha Brito, Orientador – EEEC/UFG

Examinador – INF/UFG

Examinador – EEEC/UFG

Goiânia, 20 de dezembro de 2010

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR AS TESES E DISSERTAÇÕES ELETRÔNICAS (TEDE) NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[x] Dissertação** **[ ] Tese**

2. Identificação da Tese ou Dissertação

Autor (a):	Karina Batista Duarte
E-mail:	karinaduarte@gmail.com
Seu e-mail pode ser disponibilizado na página?	[x]Sim [ ] Não
Vínculo empregatício do autor	Docente de Ensino Superior na Universidade Estadual de Goiás (UEG)
Agência de fomento:	Fundação de Amparo à Pesquisa do Estado de Goiás Sigla: FAPEG
País:	Brasil UF: GO CNPJ: 08.198.103/0001-92
Título:	Programação Genética aplicada no Processo de Descoberta de Conhecimento em Bases de Dados de Redes de Pesquisa
Palavras-chave:	Computação Evolutiva, Descoberta de Conhecimento, Mineração de Dados, Programação Genética, Redes de Pesquisa
Título em outra língua:	Genetic Programming Applied in the Process of Knowledge Discovery in Databases for Research Networks
Palavras-chave em outra língua:	Evolutionary Computation, Discovery of knowledge, Data Mining, Genetic Programming, Networks Research
Área de concentração:	Engenharia de Computação
Data defesa: (dd/mm/aaaa)	28/10/2013
Programa de Pós-Graduação:	Engenharia Elétrica e de Computação
Orientador (a):	Leonardo da Cunha Brito
E-mail:	brito.lc@gmail.com
Co-orientador (a): *	
E-mail:	

*Necessário ao CPF quando não constar no SisPG

3. Informações de acesso ao documento:

Liberação para disponibilização?* [x] total [ ] parcial

Em caso de disponibilização parcial, assinale as permissões:

[ ] Capítulos. Especifique: _____

[ ] Outras restrições: _____

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF ou DOC da tese ou dissertação.

O Sistema da Biblioteca Digital de Teses e Dissertações garante aos autores, que os arquivos contendo eletronicamente as teses e ou dissertações, antes de sua disponibilização, receberão procedimentos de segurança, criptografia (para não permitir cópia e extração de conteúdo, permitindo apenas impressão fraca) usando o padrão do Acrobat.

_____ Data: 2 / 12 / 2013

Assinatura do (a) autor (a)

*Em caso de restrição, esta poderá ser mantida por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Todos os termos e metadados ficarão sempre disponibilizados.

Ao meu pai e ao Júlio,

Meus eternos heróis.

À Deus, autor da vida e de todo o conhecimento que nela há, pela oportunidade concedida.

Ao Prof. Dr. Leonardo Guerra de Rezende Guedes, Presidente da FAPEG, por sua fundamental contribuição a este trabalho.

Ao Prof. Dr. José Clecildo Barreto Bezerra, Diretor Científico da FAPEG, amigo e grande incentivador.

Á Fundação de Amparo a Pesquisa do Estado de Goiás (FAPEG) e à Rede Goiana de Pesquisa em Computação Aplicada, pelas informações relevantes ao estudo.

A FINEP/MCT Ch 08/2005, que financiou o projeto Estruturação e Ampliação dos Sistemas Estaduais de C&T em Goiás, que deu origem a implantação das Redes Goianas de Pesquisa na FAPEG.

Ao Prof. Dr. Leonardo da Cunha Brito, pela confiança, e demais professores do programa de Mestrado da Escola de Engenharia Elétrica e de Computação da Universidade Federal de Goiás.

Aos professores do curso de Sistemas de Informação da Universidade Estadual de Goiás.

Aos amigos que me incentivaram continuamente e acreditaram em mim. Por suas sugestões, palpites e críticas.

À minha irmã Sheila, Ricardo, Felipe, Lucas e Isabela.

Com carinho especial à minha mãe, pelo apoio incondicional e por cuidar de mim.

Muito Obrigada!

“O temor do Senhor é o princípio do conhecimento.’

Proverbios 1:7

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

AG	***Algoritmo Genético***
AF	***Adjusted Fitnesss***
BI	***Business Inteligence***
C	***Linguagem de Programação C***
CNPq	***Conselho Nacional de Desenvolvimento Científico e Tecnológico***
CONFAP	***Conselho Nacional das Fundações de Amparo a Pesquisa***
C&T	***Ciência e Tecnologia***
CT&I	***Ciência, Tecnologia e Inovação***
DCBD	***Descoberta do Conhecimento em Bases de Dados***
DW	***Data Warehousing***
DM	***Data Minning***
FAPEG	***Fundação de Amparo a Pesquisa do Estado de Goiás***
FAPEGestor	**Sistema de Gestão da FAPEG**
GC	**Gestão do Conhecimento**
GP	***Genetic Programming***
GNU	***General Public Licence***
IES	***Instituição de Ensino Superior***
KDD	***Knowledge Discovery in Databases***
LISP	**Linguagem de Programação LISP**
MD	***Mineração de Dados***
MCT	***Ministério de Ciência e Tecnologia***
NF	***Normalized Fitness***
PG	***Programação Genética***
PUC	***Pentaho User Console***
PAC	***Pentaho Administration Console***
PDI	***Pentaho Data Integration***
PME	***Pentaho Metadata Editor***
PRD	***Pentaho Report Desing***

P&D	***Pesquisa e Desenvolvimento***
RF	***Raw Fitness***
R&D	***Research and Development***
SQL	***Structured Query Language***
STI	***Science, Technology and Innovation***
SGBD	***Sistemas Gerenciadores de Bancos de Dados***
SIFAPs	***Sistema de Indicadores das Fundações de Amparo a Pesquisa***

RESUMO

A Programação Genética (PG) é um algoritmo heurístico de Mineração de Dados (MD), que pode ser aplicado na tarefa de classificação. Trata-se de um método da Computação Evolutiva inspirado nos mecanismos de seleção natural, da teoria de Charles Darwin, declarada em 1859 em seu livro “A Origem das Espécies”. A partir de uma população inicial, o método busca ao longo de um conjunto de gerações a descoberta de soluções bem adaptadas ao ambiente do problema. O método de PG foi proposto por John Koza em 1990, que demonstrou em uma de suas aplicações, a indução na formação de árvores de decisão em processos de classificação de dados. Dentro deste contexto, o estudo desenvolvido neste trabalho tem como objetivo principal a investigação dos conceitos de PG e sua aplicação sobre uma base de dados de Redes de Colaboração Científica, auxiliando como ferramenta de gestão em estudos prospectivos de tendências para o estabelecimento de eixos comuns em políticas públicas de Ciência, Tecnologia e Inovação (CT&I), com foco em desenvolvimento regional. O método é aplicado sobre um conjunto de atributos, classificando-os de forma a identificar relações de similaridade entre os grupos de pesquisadores que compõem a rede. O estudo envolve conceitos de Descoberta do Conhecimento em Bases de Dados (DCBD) e Mineração de Dados (MD). As Redes de Colaboração Científica, ou Redes de Pesquisa, estão inseridas no contexto dos pequenos grupos das Redes Sociais, o ambiente é dinâmico devido à facilidade para troca de informações e articulação entre os indivíduos, favorecendo a formação de novos grupos, fato que torna ilimitado o crescimento da Rede. A combinação das características desses grupos, gerada pelos relacionamentos entre eles, configura-se como um caso de decisão multi-critério, dotando a aplicação de certa complexidade. Neste sentido, pretende-se com a aplicação do método da PG a geração de regras de classificação que levem à descoberta de grupos de pesquisadores com características similares, que em um processo planejado poderiam ser induzidos à formação de grupos fortalecidos e consolidados. O estudo contribui no sentido de explorar o potencial da Programação Genética como um algoritmo classificador, bem como, usá-lo como método na construção de ferramentas de apoio ao planejamento e tomada de decisão em CT&I.

Palavras Chave: Computação Evolutiva, Descoberta do Conhecimento, Mineração de Dados, Programação Genética, Redes de Pesquisa.

ABSTRACT

The Genetic Programming (GP) is a heuristic algorithm for Data Mining (DM), which can be applied to the classification task. This is a method of evolutionary computing inspired in the mechanisms of natural selection theory of Charles Darwin, declared in 1859 in his book "The Origin of Species." From an initial population, the method search over a number of generations to find solutions adapted to the environment of problem. The PG method was proposed in 1990 by John Koza, who demonstrated in one of its applications, the induction in formation of decision trees in the process of data classification. Within this context, the study developed in this work has as main objective the investigation of the concepts of PG and its application on a database of scientific collaboration networks, helping as a management tool in prospective studies of trends for the establishment of common axes in public policy of Science, Technology and Innovation (STI), focusing on regional development. The method is applied on a set of attributes, sorting them in order to identify similarity relationships between groups of researchers that comprise the network. The study involves the concepts of Knowledge Discovery in Databases (KDD) and Data Mining (DM). Networks of Scientific Collaboration, or Networks Research, are inserted in the context of small groups of social networks, the environment is dynamic due to the easy of information exchange and links between individuals, favoring the formation of new groups, which makes the growth of the network unlimited. "The combination of these groups, generated by the relationships between them, appears as a case of multi-criteria decision, granting the application of some complexity. In this sense, it is intended to apply the method of PG for generation of classification rules that lead to the discovery of groups of researchers with similar traits, which in a planned process could be induced to form groups strengthened and consolidated. The study helps to exploit the potential of genetic programming as a classifier algorithm, as well as use it as a method to build tools to support planning and decision making in STI.

Keyword: *Evolutionary Computation, Discovery of Knowledge, Data Mining, Genetic Programming, Networks Research.*

1 INTRODUÇÃO

O paradigma da Programação Genética (PG) foi desenvolvido por John Koza na década de 90, que demonstrou em uma de suas aplicações, a indução na formação de árvores de decisão em processos de classificação de dados. Trata-se de um método da Computação Evolutiva inspirado nos mecanismos de seleção natural, da Teoria de Charles Darwin, declarada em 1859 em seu livro "A Origem das Espécies". A Programação Genética tem sido investigada para a descoberta de regras de classificação. Este estudo é citado por (Falco, Cioppa e Tarantino, 2002, p. 258), os quais informam que heurísticas que exploram soluções de grandes dimensões, sem realizar buscas exaustivas, mas que fornecem regras de classificação inteligentes tem sido muito atraente.

A teoria de Charles Darwin (1859) parte do princípio que indivíduos na população, geneticamente mais qualificados, possuem maior probabilidade de chegar à fase adulta e de transmitir suas características genéticas para seus descendentes, aumentando essas características na população. Por analogia à teoria de Darwin (1859), o método da Programação Genética constitui-se uma combinação aleatória de partes de informação, que em um processo evolucionista vai formando novos conhecimentos sobre o objeto estudado. A regra de classificação, chamada de indivíduo, surge desse processo e vai sendo melhorada a partir das características transmitidas por regras antecessoras, até que a melhor regra tenha características dominantes sobre as demais. Após determinada a regra de classificação, esta é aplicada sobre um conjunto de dados, classificando de forma a explicitar o conhecimento descoberto sobre os dados estudados.

O estudo de caso selecionado para a aplicação do método de PG se refere à base de dados da Fundação de Amparo a Pesquisa do Estado de Goiás (FAPEG), uma agência governamental de fomento à pesquisa científica, que concede os recursos captados para a pesquisa, através de um modelo de rede de colaboração científica, denominado Redes Goianas de Pesquisa. O conhecimento em rede acaba por gerar um imenso volume de dados, por isso, o grande desafio é torná-los úteis, convertendo-os em informações que por sua vez gerem novos conhecimentos.

As Redes de Pesquisa estão inseridas no contexto dos pequenos grupos das redes sociais, o ambiente é dinâmico devido à facilidade para troca de informações e articulação entre os indivíduos, favorecendo a formação de novos grupos, incrementando o crescimento da Rede. A combinação das características desses grupos, gerada pelos relacionamentos entre eles, configura-se como um caso de decisão multi-critério, dotando a aplicação de certa complexidade. Neste sentido, pretende-se com a utilização do método da PG, a geração de regras de classificação que levem à descoberta de grupos de pesquisadores com características similares, que em um processo planejado poderiam ser induzidos à formação de grupos fortalecidos e consolidados. Esta aplicação poderá auxiliar como ferramenta de gestão em estudos prospectivos de tendências para o estabelecimento de eixos comuns em políticas públicas de Ciência, Tecnologia e Inovação (CT&I), voltadas para o desenvolvimento regional.

O processo de investigação do estudo envolve conceitos de Descoberta do Conhecimento em Bases de Dados (DCBD) e Mineração de Dados (MD), uma fase do processo de DCBD, sendo composta por várias técnicas, como classificação, associação e agrupamento, e algoritmos dentre os quais estão os métodos de computação evolucionária tais como “Algoritmos Genéticos” e “Programação Genética”.

O presente trabalho está sistematizado em cinco Capítulos. Após este Capítulo introdutório, o Capítulo II analisa a literatura pertinente, discorrendo sobre os conceitos de Computação Evolucionária, Algoritmos Genéticos e Programação Genética; A tarefa de Classificação usando PG; O processo de DCBD e a fase de Mineração de Dados; A descrição do estudo de caso finaliza este Capítulo. O Capítulo III descreve os materiais e métodos científicos usados. O Capítulo IV descreve os resultados obtidos através da aplicação de cada uma das fases do processo de DCBD, com especial destaque à aplicação do método de Programação Genética na fase de Mineração de Dados. O Capítulo V é reservado às considerações finais, conclusões e recomendações de trabalhos futuros.

O estudo contribui no sentido de explorar o potencial da Programação Genética como um algoritmo classificador, bem como, usá-lo como método na construção de ferramentas de apoio ao planejamento e tomada de decisão, ao se estabelecer ações e metas para gestão das Redes de Colaboração Científica e indução ao desenvolvimento de CT&I.

2 REVISÃO DE LITERATURA

2.1 CONTEXTUALIZAÇÃO DA PROGRAMAÇÃO GENÉTICA

A Computação Evolucionária é uma área da ciência da computação inspirada nos princípios da evolução natural, introduzida por Charles Darwin em 1859 em seu livro “A Origem das Espécies”. Os princípios da Teoria da Evolução são usados para pesquisar soluções aproximadas para a solução de problemas usando o computador. O principal requisito para a aplicação da computação evolucionária é que a qualidade da possível solução possa ser calculada. Assim, poderia ser possível classificar algumas possíveis soluções visando à qualidade da solução e se a solução resolve o problema (Eggermont, 2005, p. 1).

A Programação Genética (PG) e o Algoritmo Genético (AG) são métodos da Computação Evolucionária que aplicam os conceitos da biologia genética, baseados na teoria da seleção natural. O paradigma da PG foi desenvolvido por John Koza na década de 90 com base nos trabalhos de John Holland na década de 70, e podem ser visto como uma extensão dos algoritmos genéticos. As principais diferenças dizem respeito ao método usado na seleção e nos operadores genéticos.

2.2 CONCEITOS GENÉRICOS DE BIOLOGIA GENÉTICA

A literatura apresenta alguns conceitos sobre biologia genética (Salman, 2007, p. 8-25), (Romão, Freitas e Pacheco, 2000, p. 5) e (Romero e Mantovani, 2004, p. 15-16), os quais são relacionados a seguir e formam os princípios básicos dos algoritmos evolucionários.

*Molécula de DNA***:** Informações genéticas dos indivíduos, que ficam armazenadas numa molécula chamada de DNA ou ADN, que significa ácido desoxirribonucléico (Salman, 2007, p. 8).

Cromossomo: A molécula de DNA é organizada e armazenada em estruturas denominadas de cromossomos que ficam no núcleo das células. Cada cromossomo apresenta-se em pares denominados de homólogos, um é herdado do pai e o outro da mãe. Todos os pares de cromossomos encontrados no núcleo de uma célula são denominados de autossomos. Cada par de cromossomos homólogos apresenta o mesmo comprimento e forma (exceto pelos

dois cromossomos sexuais que determinam o sexo) e contém a informação genética para a mesma característica. O número de pares de cromossomos é típico de cada espécie (Salman, 2007, p. 8).

Genoma: É todo o material genético de uma célula. Um genoma pode ter vários cromossomos (Salman, 2007, p. 9).

Gene: Segmento de DNA situado numa posição específica de um determinado cromossomo e que participa da manifestação fenotípica de certa característica. O local específico de um gene no cromossomo é chamado de loco e o plural de locos (Salman, 2007, p. 10). O gene corresponde a cada elemento do cromossomo, (Romero e Mantovani, 2004, p. 15).

Alelo: Tipo específico de informação existente no gene (Romero e Mantovani, 2004, p. 15).

Genótipo: É a combinação de genes alelos proveniente das células germinativa feminina e masculina. Por exemplo, para o gene A que possui um alelo a, existem três genótipos possíveis de serem formados quando há a união dos gametas: (AA, Aa e aa). O genótipo é essencialmente uma característica fixa de um organismo, fica constante durante toda a vida e não é mudada por fatores do ambiente (Salman, 2007, p. 11). A constituição hereditária de um indivíduo é dada pelo seu genótipo (Romão, Freitas e Pacheco, 2000, p. 5).

Fenótipo: Característica visual ou seletiva no indivíduo (Romero e Mantovani, 2004, p. 15). Para algumas características, o fenótipo muda continuamente durante a vida de um indivíduo em resposta a fatores ambientais (Salman, 2007, p. 11).

População Genética: Grupos de indivíduos de mesma espécie que se reproduzem entre si e por isso apresentam propriedades comuns numa dimensão de espaço e tempo (Salman, 2007, p. 15).

Hereditariedade: É a transmissão de características dos pais para seus filhos por meio do material genético, durante o processo da reprodução (Salman, 2007, p. 13).

Reprodução (Recombinação Genética): Ocorre quando dois cromossomos gêmeos se separam no processo de divisão celular de uma célula reprodutiva. Assim, seja um cromossomo "A" é herdado do pai e outro cromossomo "B" da mãe. Quando a célula reprodutiva se separa e se duplica, cada gameta gerado tem parcelas do cromossomo "A" e "B". O resultado desse processo é a produção de gametas que possuem parcelas dos

cromossomos gêmeos "A" e "B" numa forma diversificada e diferente umas das outras (Romero e Mantovani, 2004, p. 16).

Seleção: Competição ou processo de sobrevivência de um indivíduo, proporcional a sua capacidade de adaptação (Romão, Freitas e Pacheco, 2000, p. 5). Existem duas formas: Natural e Artificial. A seleção natural consiste em selecionar genótipos mais bem adaptados a uma determinada condição de ambiente, eliminando aqueles desvantajosos para essa mesma condição. Na seleção artificial um conjunto de requisitos é imposto pelo homem para dirigir a probabilidade de um indivíduo se reproduzir (Salman, 2007, p. 18).

Mutação: São alterações que ocorrem ao acaso no material genético de um indivíduo e que são transmitidas à descendência podendo, em alguns casos, provocar alterações fenotípicas (Salman, 2007, p. 25). A mutação é fonte de diversidade genética (Romero e Mantovani, 2004, p. 15). Ela possibilita a geração de genes não presentes na população corrente (Romão, Freitas e Pacheco, 2000, p. 5).

Adaptação: Valor adaptativo também denominado "*fitness*", que se refere à maior probabilidade de, em um determinado ambiente, um indivíduo deixar descendentes. O valor de "*fitness"* determina as chances de sobrevivência e reprodução do indivíduo (Salman, 2007, p. 18).

Migração: É a mudança de indivíduos de uma população que está se reproduzindo para outra população que também se reproduz (Salman, 2007, p. 24).

2.3 ALGORITMO GENÉTICO (AG)

O Algoritmo Genético foi concebido e desenvolvido por John Holland na década de 70, sendo popularizado por David Goldberg, em 1989.

> O AG apresenta um grupo de soluções candidatas, denominado população, na região de soluções. Através de mecanismos como a seleção natural e o uso de operadores genéticos, tais como a mutação e o cruzamento, os cromossomos com melhor aptidão são encontrados. A seleção natural garante que os cromossomos mais aptos gerem descendentes nas populações futuras. Usando um operador de cruzamento, o AG combina genes de dois ou mais cromossomos de pais previamente selecionados para formar novos cromossomos, os quais têm grande possibilidade de serem mais aptos que os seus genitores. (Dantas, 2008, p. 58)

Definições básicas usadas nos Algoritmos Genéticos, encontradas na literatura, serão relacionadas a seguir, conforme (Lacerda e Carvalho, 1999, p. 103), (Castro, 2001, p. 36-39), (Romão, Freitas e Pacheco, 2000, p. 5-7), (Carvalho, 2005, p. 18-22) e (Romero e Mantovani, 2004, p. 17-22).

Algoritmo: Segundo (Romero e Mantovani, 2004, p. 17), um algoritmo genético elementar realiza a seguinte seqüência de operações:

> Passo 1: Gerar a população inicial após escolher o tipo de codificação do cromossomo;
>
> Passo 2: Calcular a função objetivo de cada indivíduo da população;
>
> Passo 3: Implementar a Seleção;
>
> Passo 4: Implementar a Recombinação (Cruzamento).
>
> Passo 5: Implementar a Mutação
>
> Passo 6: Avaliar os indivíduos sobreviventes
>
> Passo 7: Se o critério de parada não for satisfeito, repetir os passos dois a seis.

Genoma e Cromossomo: Nos Algoritmos Genéticos, o genoma e o cromossomo representam a estrutura de dados que codifica a solução para o problema. Cada cromossomo representa deste modo, uma solução do problema (Lacerda e Carvalho, 1999, p. 103).

Gene: É a unidade básica do cromossomo. Cada cromossomo tem certo número de genes, cada um descrevendo determinada variável do problema (Castro, 2001, p. 36). Corresponde a um elemento da estrutura de dados que representa o cromossomo (Lacerda e Carvalho, 1999, p. 103).

Genótipo e Fenótipo: Representam a informação contida no cromossomo (Lacerda e Carvalho, 1999, p. 103).

Alelo: Representa os valores que o gene pode assumir (Lacerda e Carvalho, 1999, p. 103).

População: Conjunto de cromossomos ou soluções (Castro, 2001, p. 36).

Indivíduo: Um simples membro da população (Lacerda e Carvalho, 1999, p. 103). Nos AGs, os indivíduos representam os cromossomos (Romão, Freitas e Pacheco, 2000, p. 5).

Geração: O número da iteração que o AG executa (Castro, 2001, p. 36). A cada nova geração um novo conjunto de indivíduos é criado usando partes dos melhores indivíduos da geração anterior (Carvalho, 2005, p. 19).

Espaço de Busca: Conjunto, espaço ou região que compreende as soluções possíveis ou viáveis do problema a ser otimizado (Castro, 2001, p. 37). Se o cromossomo representa

“n” parâmetros de uma função, então o espaço de busca é um espaço com “n” dimensões (Lacerda e Carvalho, 1999, p. 91).

Função Objetivo ou de Avaliação (Function Fitness): É a função que se quer otimizar. Ela contém a informação numérica do desempenho de cada cromossomo na população, indicando a qualidade da solução candidata avaliada (Castro, 2001, p. 37). O processo de seleção é diretamente influenciado pela função de fitness (Carvalho, 2005, p. 20).

Operações Genéticas: Operações que o Algoritmo Genético realiza sobre cada um dos cromossomos (Castro, 2001, p. 37). Os operadores genéticos de seleção, cruzamento e mutação fornecem o mecanismo básico de busca e são usados para criar novas soluções baseadas nas melhores soluções existentes na população atual do AG (Romão, Freitas e Pacheco, 2000, p. 6).

Seleção: Operação responsável por selecionar dois indivíduos, dentre os que obtiveram melhor avaliação da função de aptidão (*Fitness)*, para serem usados pelos operadores de cruzamento e mutação, a fim de gerar a próxima geração (Romão, Freitas e Pacheco, 2000, p. 7). Existem alguns métodos que ajudam no processo: Seleção Proporcional, Torneio e a Estratégia Elitista.

Seleção Proporcional: Também conhecido com Método da Roleta. Seu funcionamento se assemelha a uma roleta, onde cada indivíduo da população é representado proporcionalmente ao seu índice de aptidão (Castro, 2001, p. 45). A probabilidade de um indivíduo ser escolhido para reprodução é diretamente proporcional ao seu valor de *fitness*, em relação à soma dos valores de *fitness* de todos os indivíduos da população, isto é, pela aptidão normalizada (Carvalho, 2005, p. 20).

Método de Torneio: Consiste em obter aleatoriamente “k” indivíduos da população, e aquele que apresentar o maior valor de *fitness* é selecionado para a reprodução (Carvalho, 2005, p. 21).

Estratégia Elitista: O objetivo é manter boas soluções na população por mais de uma geração. Nesta estratégia são selecionados os “n” melhores indivíduos da geração atual para a próxima geração, preservando assim, os melhores indivíduos da população (Carvalho, 2005, p. 21). O método busca a convergência da melhor solução (Castro, 2001, p. 50).

Cruzamento (Recombinação): Operação que realiza a troca de material genético entre os indivíduos selecionados para a reprodução (Romão, Freitas e Pacheco, 2000, p. 7). Nem sempre todos os indivíduos são submetidos à recombinação, em geral existe uma taxa de

cruzamento que controla a freqüência com que os cruzamentos ocorrerão (Carvalho, 2005, p. 22).

Mutação: Operação que realiza alteração em algum gene escolhido aleatoriamente (Romão, Freitas e Pacheco, 2000, p. 7). O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população. O operador de mutação é aplicado aos indivíduos através de uma taxa de mutação geralmente pequena (Carvalho, 2005, p. 22).

Existem inúmeras formas de representação das variáveis do cromossomo, tais como: binária, números inteiros, números reais ou um conjunto de regras de soluções. No contexto da Mineração de Dados a codificação de um indivíduo é, em geral, uma seqüência linear de condições de regras, sendo geralmente cada condição um par atributo-valor (Carvalho, 2005, p. 20).

Os AGs são métodos naturais para implementação de buscas multiobjetivo, em que se deseja encontrar múltiplas soluções ótimas. O método pode ser aplicado durante o processo de DCBD, tanto na fase de pré-processamento, para seleção de atributos, como na fase de pós-processamento, para interpretação dos resultados.

Conforme descrito por (Pappa, 2002, p. 41), os algoritmos genéticos vem sendo aplicados na solução de problemas em que o espaço de busca é grande.

No trabalho "*Genetic clustering of social networks using random walks*", relacionado ao uso de Algoritmos Genéticos para Agrupamentos, os autores (Firat, Chatterjee e Mustafa, 2007, p. 1), defendem o uso de algoritmos genéticos como técnica heurística de pesquisa para agrupamento em redes sociais. Eles informam que o uso de técnicas de força bruta se torna rapidamente impraticável em redes de tamanho incremental. Assim as soluções heurísticas são necessárias para obter soluções aproximadas.

Outro trabalho que vale a pena ser mencionado, no qual foram usados algoritmos genéticos é: "*A method for member selection of R&D teams using the individual and collaborative information*", dos autores (Fan, Feng, *et al.*, 2009, p. 2). A justificativa é que a seleção de equipes de pesquisa e desenvolvimento é um problema complexo de decisão, o qual precisa considerar múltiplos critérios. Os autores propõem um método de seleção de equipes de pesquisa e desenvolvimento, em que informações individuais dos membros e informações colaborativas entre os membros são consideradas. Para resolver o modelo, um algoritmo genético multiobjetivo foi desenvolvido.

2.4 PROGRAMAÇÃO GENÉTICA (PG)

O paradigma da Programação Genética foi desenvolvido por John Koza na década de 90 com base nos trabalhos de John Holland, que desenvolveu a teoria dos Algoritmos Genéticos. O método da PG foi descrito em (Koza, 1992), sendo alguns princípios relacionados a seguir.

> PG evolui uma população de programas de computador, que são possíveis soluções para um problema de otimização, usando o princípio darwinista da sobrevivência dos mais aptos. Ela usa as operações biologicamente inspiradas, como reprodução, cruzamento e mutação. Cada programa ou indivíduo na população é geralmente representado como uma árvore composta de funções e terminais adequados para o domínio do problema. (Muni, Pal e Das, 2004, p. 184)

A Figura 1 mostra os cinco principais passos preparatórios para a versão básica da PG. As etapas preparatórias (mostrada no topo da figura) são as entradas, o programa de computador (mostrado na parte inferior), criado automaticamente pela PG pode resolver o problema do usuário e corresponde à saída do sistema.

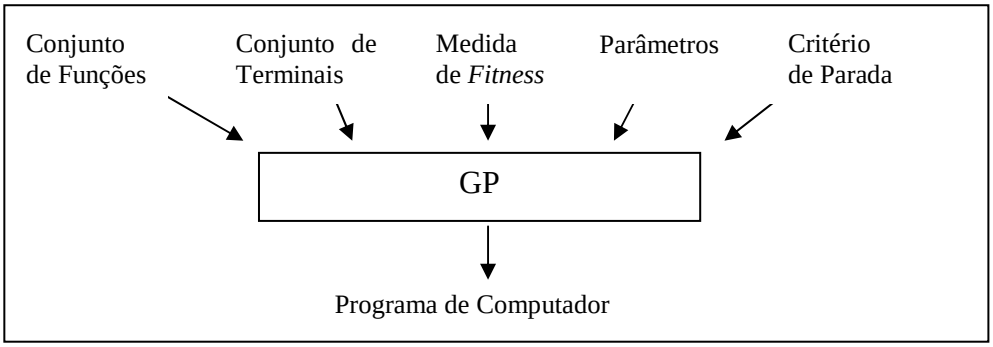


Figura 1 – Passos Preparatórios para a Versão Básica da PG.
Fonte: Adaptado de (Koza, 2003, p. 47)

Um *programa de computador* é composto de entrada, processamento e saída, parte do princípio que problemas de otimização podem ser rapidamente resolvidos se existir soluções de programas de computador disponíveis para um problema específico, que produza a saída desejada para entradas particulares, conforme demonstrado na Figura 2.

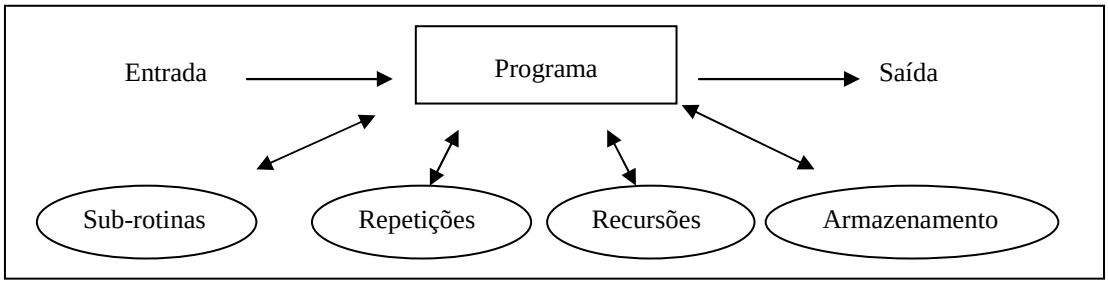


Figura 2 – Programa de Computador
Fonte: Adaptado de (Koza, 2003, p. 66)

Os programas computacionais, independentemente da linguagem em que são codificados, podem ser vistos como uma seqüência de aplicações de funções (operações) e argumentos (valores). O Quadro 1 apresenta o pseudocódigo de um programa escrito na linguagem de programação “C”. O Quadro 2 mostra as entradas e saídas do programa em linguagem “C”, do Quadro 1. A Figura 3 mostra a árvore de decisão correspondente ao programa em Linguagem “C”, do Quadro 1. O Quadro 3 apresenta a árvore de decisão correspondente à Figura 3 em expressão simbólica pré-fixada.

Quadro 1 – Pseudocódigo de um Programa em Linguagem "C"

```
int foo (int time) {
      int temp1, temp2;
      if (time > 10)
         temp1 = 3;
      else
         temp1 = 4;
      end
      temp2 = temp1 + 2;
      return (temp2);
end}
```

Fonte: Adaptado de (Koza, 2003, p. 55)

Quadro 2 – Entradas e Saídas de um Programa em Linguagem “C”

Input (time)	Output
0	6
1	6
2	6
3	6
4	6
5	6

Fonte: Adaptado de (Koza, 2003, p. 56)

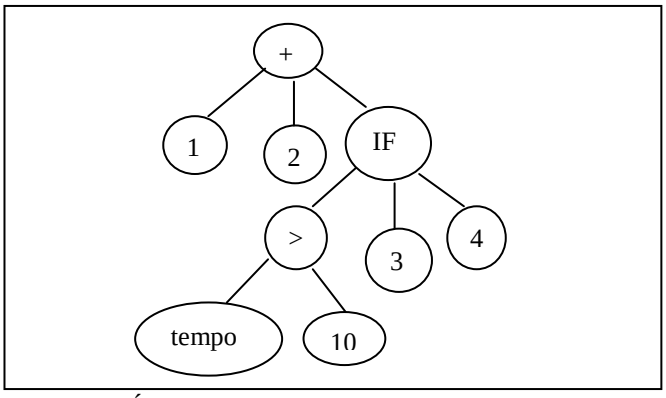


Figura 3 – Árvore de decisão de um Programa em “C”
Fonte: Adaptado de (Koza, 2003, p. 57)

Quadro 3 – Notação Pré-fixada da Árvore do Programa em Linguagem “C”

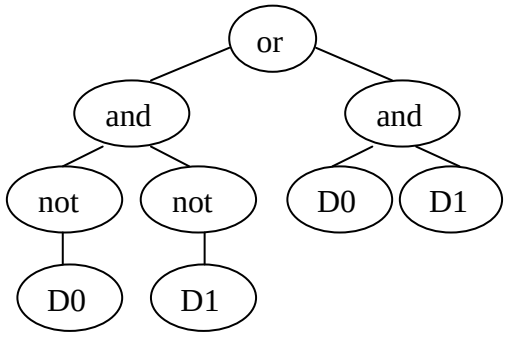
(+ 1 2 (IF (> TIME 10) 3 4))

Fonte: Adaptado de (Koza, 2003, p. 57)

Os tipos de *estruturas de dados* mais utilizadas em PG são: listas, pilhas, filas, árvores ou grafos. Estas estruturas são organizadas sobre o conceito de “nós”, que podem ser divididos em um ou mais campos de informação, dependendo do “nó” ser do tipo primitivo ou abstrato. A seleção de qual estrutura de dados é a mais adequada para a representação da informação depende principalmente dos tipos de operações que precisam ser realizadas e da linguagem de programação que será usada. A estrutura de dados representa a estrutura do cromossomo. Cada gene é representado por uma informação que pode ser um atributo ou o valor do atributo.

A *Linguagem de Programação LISP ou LISt Processing*, concebida por John McCarthy em 1958, tornou-se a principal linguagem dos estudos em Inteligência Artificial. A lista é a estrutura de dados fundamental da linguagem. Cada programa em LISP é portanto, uma expressão. As expressões são de tamanho indefinido e tem uma estrutura de árvore binária. Esta é a linguagem original da programação genética. Em LISP, existem apenas dois tipos de entidades: átomos (constantes e variáveis) e listas (conjunto ordenado de itens delimitados por um par de parênteses). Uma expressão simbólica (*S-expression*): é um átomo ou uma lista. Esta é a única forma sintática existente, ou seja, todos os programas em LISP são expressões simbólicas e permitem codificar todos os tipos de construções admitidas para um programa computacional (seqüencia, seleção e repetição), com base na notação pré-fixada. O Quadro 4 apresenta as três formas de representação da linguagem LISP.

Quadro 4 – Formas de Representação da Linguagem LISP

S-expression:	**Árvore Binária**
(OR (AND (NOT D0) (NOT D1)) (AND D0 Dl))	
Pseudocódigo: (defun gt (primeiro-argumento segundo-argumento) "Maior valor numérico da função" (if (>primeiro-argumento segundo-argumento) 1 -1))).	

Nota: Adaptado de (Koza, 1992, p. 81-83)

Para avaliar uma expressão ou lista, assume-se que o primeiro elemento é uma função, sendo os elementos seguintes, seus argumentos. Portanto é possível tratar os programas diretamente como se fossem dados, facilitando a aplicação da programação genética.

O *algoritmo da PG* é simples, semelhante ao do AG e pode ser resumido como mostrado a seguir:

1. Gerar uma população inicial de composição randômica de funções e terminais do problema (programas de computador);
2. Iterativamente realizar os seguintes passos até que o critério de terminação seja satisfeito;
   a) Executar cada programa da população e atribui-lhe um valor de *fitness*, que expressa o quanto ele resolve o problema. (Isto é, o quanto o programa está próximo da solução ideal);
   b) Criar uma nova população de programas de computador aplicando as seguintes operações primárias. As operações são aplicadas aos programas de computador na população escolhida com probabilidade baseada na *fitness*:
      i. Copiar os programas de computador existentes para a nova população.
      ii. Criar novos programas de computador pela recombinação genética de partes escolhidas de dois programas existentes.
3. O melhor programa de computador que apareceu em qualquer geração é designado como o resultado da programação genética. Este resultado pode ser uma solução (ou uma solução aproximada) para o problema.

O *Espaço de Busca na PG* é o conjunto de estruturas possíveis que se submetem a adaptação do paradigma de programação genética. Trata-se do conjunto de todas a possíveis combinações de funções, que podem ser compostas recursivamente a partir do conjunto de funções F = {f1, f2, ... , fn} e o conjunto de terminais T = {a1, a2, ... , am}. Cada função particular $f \in F$ requer um número z(f) de argumentos $z(f1)$, $z(f2)$ ..., $z(fn)$. Isto é, a função *fi* tem aridade $z(fi)$. As funções podem conter operadores aritméticos (+,−,*, /), funções matemáticas (seno, exp, log, etc), operadores lógicos (e,ou,não, etc), operadores condicionais tais como If-Then-Else, operadores de repetição tais como Do-Until, funções recursivas e funções de domínio específico do problema. Os terminais podem ser variáveis de argumentos atômicos, constantes de argumentos atômicos, como zero e um, e em alguns casos, podem ser outras entidades atômicas como funções sem argumentos.

John Koza estabeleceu duas propriedades para as funções e terminais, sendo denominadas:

a) *Propriedade de Fechamento*, em que cada função do conjunto *F* deve aceitar como seus argumentos, qualquer valor que possa ser retornado por qualquer

função ou terminal. Ela demanda que o conjunto de funções seja muito bem definido, de forma a fechar as combinações de argumentos que possam ser encontradas.

b) *Propriedade de Suficiência*, que visa garantir a convergência para uma solução. Deve existir uma forte evidência de que alguma composição de funções e terminais possa produzir uma solução. O conjunto de funções e o conjunto de terminais devem ser capazes de expressar a solução do problema.

A geração de uma *população inicial* começa pela seleção aleatória de uma das funções de $f \in F$, para ser a raiz da árvore. Para cada um dos argumentos de *f*, escolhe-se aleatoriamente um elemento de $F \cup T$ O processo prossegue até que se tenham apenas nós terminais como folhas da árvore. Geralmente é especificado um limite máximo para a profundidade da árvore, para se evitar árvores muitos grandes. Os principais métodos para geração da população inicial são: *Full, Grow ou Ramped-half-and-half.*

a) O Método Full envolve a criação de árvores cujos ramos terão a mesma profundidade. Isto é feito através da seleção de funções para os nós cuja profundidade seja inferior a desejada e a seleção de terminais para os nós de profundidade máxima. O Quadro 5 apresenta um algoritmo simples do Método *Full*, segundo (Dolan, 2009).

Quadro 5 – Algoritmo Método Full

```
public GPNode generateFull(int maxDepth) {
        GPNode root;
        //Se não é profundidade máxima, escolhe uma função
        if(maxDepth > 1)
                root = getFunction();
        //Senão, escolhe um terminal
        else
                root = getTerminal();
        end
        //Recursivamente atribui nós filhos
        for(int i = 0; i < root.numChildNodes(); i++)
                root.setChildNode(i, generateFull(maxDepth - 1));
        end
        return root;                }
```

Fonte: (Dolan, 2009).

b) O *Método Grow* envolve a criação de árvores cuja profundidade é variável. A escolha dos nós é feita aleatoriamente, do conjunto união entre funções e terminais $C = F \cup T$, respeitando-se uma profundidade máxima. O Quadro 6 apresenta o algoritmo do Método Grow, segundo (Dolan, 2009).

Quadro 6 – Algoritmo Método Grow

```
public GPNode generateGrow(int maxDepth) {
        GPNode root;
        if(maxDepth > 1)
                root = getNode();
        //Se é profundidade maxima escolhe um terminal
        else
                root = getTerminal();
        end
        ////Recursivamente atribui nós filhos
        for(int i = 0; i < root.numChildNodes(); i++)
                root.setChildNode(i, generateGrow(maxDepth - 1));
        end
        return root; }
```

Fonte: (Dolan, 2009).

c) O *Método Ramped-half-and-half* é uma combinação dos métodos *Full* e *Grow*, gerando árvores dos mais variados tamanhos e formas. A quantidade de árvores geradas para cada método é especificada podendo, por exemplo, gerar 50% de árvores pelo método *Full* e 50% pelo método *Grow*.

A *Função de Avaliação ou Aptidão* consiste em que a cada indivíduo da população (programa de computador) seja atribuído um valor de *fitness*, como resultado de sua interação com o meio ambiente. Os valores de entrada do programa são confrontados com os valores de resposta esperados para a saída, quanto mais próximos do resultado desejado, melhor é o programa. A função *fitness* determina o quanto um programa será capaz de resolver o problema. Uma dificuldade é encontrar a função *fitness* adequada. A *fitnesss* pode ser medida de várias maneiras, os métodos mais comuns são: *raw fitness, standardized fitness, adjusted fitness e normalized fitness.*

a) *Raw Fitness (Aptidão Nata):* Número de casos de treinamento classificados dentro das classes corretas.

b) *Standardized Fitness (Aptidão Padronizada):* Número total de casos de treinamento menos o *Raw Fitness.*

c) *Adjusted Fitness* (Aptidão Ajustada): É obtida através da aptidão padronizada. A aptidão ajustada varia entre zero (0) e um (1), sendo que os maiores valores representam os melhores indivíduos. Se $s(i,t)$ representa a aptidão padronizada do indivíduo *i* na geração *t*, então a aptidão ajustada $a(i,t)$ é calculada pela fórmula (1):

$$a(i,t) = \frac{1}{1 + s(i,t)} \qquad (1)$$

d) *Normalized Fitness* (Aptidão Normalizada): Se $a(i,t)$ representa a aptidão ajustada do indivíduo i na geração t, então a aptidão normalizada $n(i,t)$ é calculado pela fórmula (2):

$$n(i,t) = \frac{a(i,t)}{\sum_{k=1}^{m} a(k,t)} \qquad (2)$$

John Koza descreve duas operações primárias principais: seleção e cruzamento; e cinto operações secundárias: mutação, edição, permutação, destruição e encapsulamento, usadas para modificar estruturas sob adaptação em PG.

A *operação de seleção*, em relação ao conceito biológico, corresponde ao processo de reprodução assexuada, que seleciona um indivíduo conforme sua aptidão e o copia para a nova geração sem modificá-lo. Existem diferentes métodos de seleção baseados no valor da *fitness*, os mais comuns são: Seleção Proporcional, Seleção por Torneio e Seleção Elitista. Estes métodos foram descritos no estudo de Algoritmo Genético. A seleção dá aos bons indivíduos a chance de serem preservados.

A operação de cruzamento (*Crossover*), em relação ao conceito biológico, corresponde ao processo de reprodução sexuada. Dois programas são selecionados e recombinados para gerar outros dois programas a partir de seus valores de *fitness*. Um ponto aleatório de cruzamento é escolhido em cada programa-pai e as árvores abaixo destes pontos são trocadas. No algoritmo genético convencional pode ocorrer o problema de convergência prematura, caso de cruzamentos em indivíduos idênticos, a população converge para um resultado globalmente sub-otimo, em que não é possível encontrar um ótimo global. No caso da Programação Genética, quando este caso ocorre, os dois descendentes resultantes, em

geral, são diferentes. Assim a convergência prematura é improvável na PG. A Figura 4 apresenta graficamente a operação de cruzamento entre pais diferentes e a Figura 5 apresenta graficamente a operação de cruzamento entre pais idênticos.

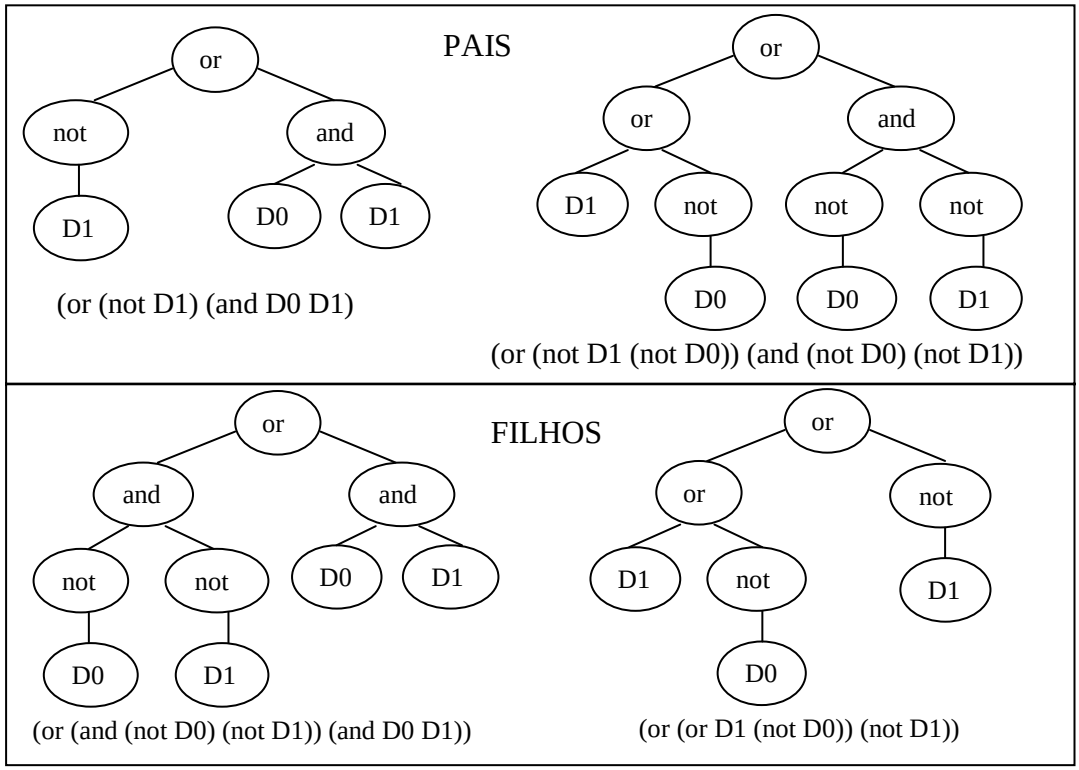


Figura 4 – Operação de Cruzamento com Pais Diferentes
Fonte: Adaptado de (Koza, 1992, p. 102)

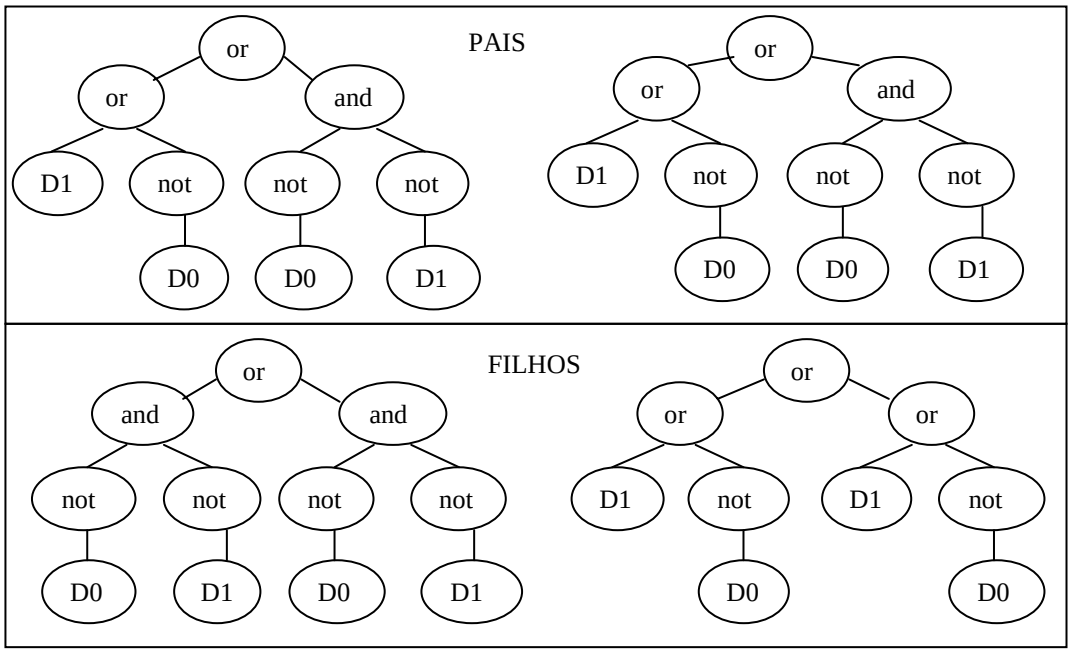


Figura 5 – Operação de Cruzamento com Pais Idênticos
Fonte: Adaptado de (Koza, 1992, p. 103)

O *operador de mutação* modifica aleatoriamente alguma característica do indivíduo. Ele é necessário para a introdução e manutenção da diversidade genética da população. A operação começa selecionando um ponto aleatoriamente dentro da árvore. Este ponto pode ser interno (função) ou externo (terminal). Em seguida o ponto selecionado e o que está abaixo dele são removidos e é inserida uma subarvore gerada aleatoriamente neste ponto. Na PG a mutação é desnecessária, pois é relativamente rara a perda da diversidade, além disso, a operação de cruzamento substitui a mutação. Por isso é considerada uma operação secundária. A Figura 6 mostra a operação de mutação.

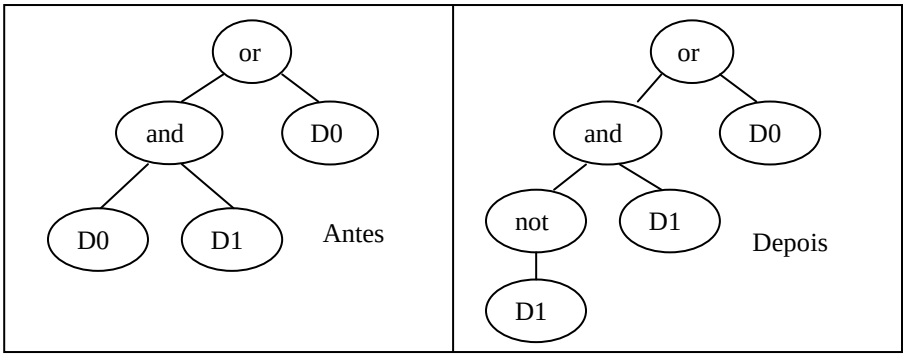


Figura 6 – Operação de Mutação
Fonte: Adaptado de (Koza, 1992, p. 106)

A operação de *Edição* é usada para simplificar expressões. Pode ser aplicada tanto para facilitar a visualização dos programas, como na execução, para reduzir o tamanho dos programas. A desvantagem deste operador é que consome muito tempo de processamento. Se uma função tem apenas constantes como argumentos, a operação de edição irá avaliá-la e substituí-la com o valor obtido a partir da avaliação. Exemplo: a expressão numérica (+ 1 2) pode ser trocada por 3 e a expressão lógica (AND T T) pode ser trocada por T (verdadeiro).

A operação de *Permutação* é também conhecida como inversão, reordenando os argumentos da função encontrados entre dois pontos selecionados de um único indivíduo, conforme Figura 7.

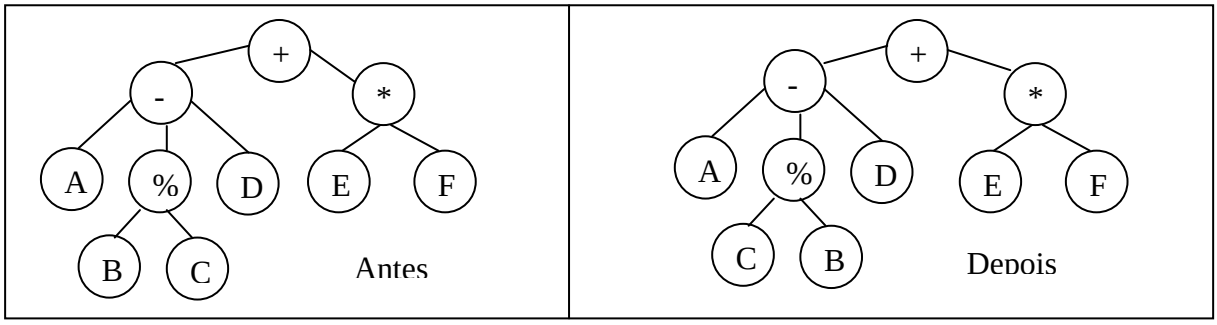


Figura 7 – Operação de Permutação
Fonte: Adaptado de (Koza, 1992, p. 108)

A operação de *Destruição* é usada para reduzir o número de indivíduos nas primeiras gerações. Essa redução é baseada no valor de aptidão de cada indivíduo, nesse caso é estabelecido um percentual que determina quantos indivíduos deve permanecer na população, o restante é eliminado pelo operador de destruição. Esta operação geralmente é aplicada na primeira geração.

A operação de E*ncapsulamento* é um meio para identificar automaticamente uma subárvore potencialmente útil, e dar-lhe um nome que possa ser referenciado e utilizado posteriormente. Nesse caso, seleciona-se um indivíduo, conforme seu valor de aptidão, seleciona-se um ponto na estrutura deste indivíduo, a sub-árvore formada a partir deste ponto é removida e a ela é dado um nome, formando uma nova função E0, E1, E2, E3, ..., En. Na estrutura do indivíduo é inserida uma referência a essa nova função. A Figura 8 mostra a operação de encapsulamento.

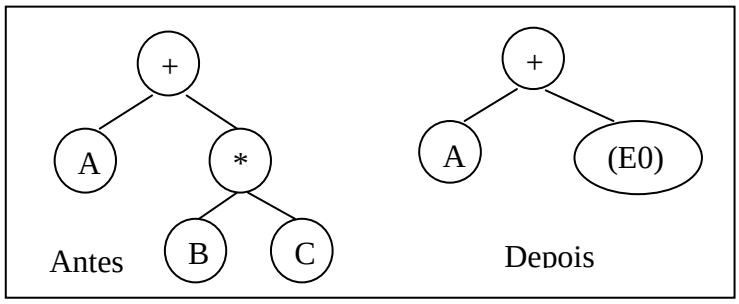


Figura 8 – Operação de Encapsulamento
Fonte: Adaptado de (Koza, 1992, p. 111)

A execução do algoritmo de programação genética geralmente termina quando um *critério de terminação* é satisfeito. O mais comum é limitar o número máximo de gerações ou executar até que uma solução satisfatória seja encontrada.

Na formação da estrutura da árvore de programas, pode ser necessário o uso de algumas *restrições sintáticas*. Por exemplo: O nó-raiz da árvore deve ser obrigatoriamente uma função que contenha o operador condicional IF; A função XSEN é definida como tendo dois argumentos (arg1 e arg2) e seu cálculo é arg1*sin(arg2*x).

2.5 PROCESSO DE DESCOBERTA DO CONHECIMENTO EM BASES DE DADOS (DCBD)

Para que o método de PG possa ser usado em uma determinada aplicação é necessário que o conjunto de dados esteja preparado com dados concisos e de qualidade. Eles são selecionados de várias fontes, pré-processados e transformados, formando um único

conjunto de dados. Após esta preparação, métodos e técnicas são aplicados com a finalidade de se extrair conhecimento útil, e é nesta fase, denominada de Mineração de Dados, que o método de PG é aplicado na tarefa de classificação. Na última fase os resultados são interpretados, explicitando o conhecimento descoberto. As fases desse processo compõem a DCBD.

A DCBD tem a função de identificar padrões válidos em dados armazenados, através do uso de técnicas e ferramentas que auxiliam na extração de conhecimento útil. A DCBD é também conhecida no termo em inglês Knowledge Discovery in Databases (KDD). O termo KDD foi estabelecido pela primeira vez em um *workshop* em 1989 para enfatizar que conhecimento é o produto final de uma descoberta baseada em dados (*datadriven*).

> Enquanto algumas pessoas tratam a Mineração de Dados como sinônimo de DCBD, outras vêem como um passo particular no processo, envolvendo a aplicação de algoritmos específicos para a extração de padrões de dados. Os passos adicionais no processo de DCBD, tais como a preparação dos dados, seleção dos dados, limpeza nos dados, incorporação de conhecimento prévio adequado e interpretação dos resultados da mineração, asseguram que o conhecimento útil seja derivado dos dados. (Mitra e Acharya, 2003, p. 4)

O processo de DCBD foi modelado por (Fayyad, Piatetsky-Shapiro e Smyth, 1996), apresentando a Mineração de Dados como uma etapa do processo, conforme apresentado na Figura 9.

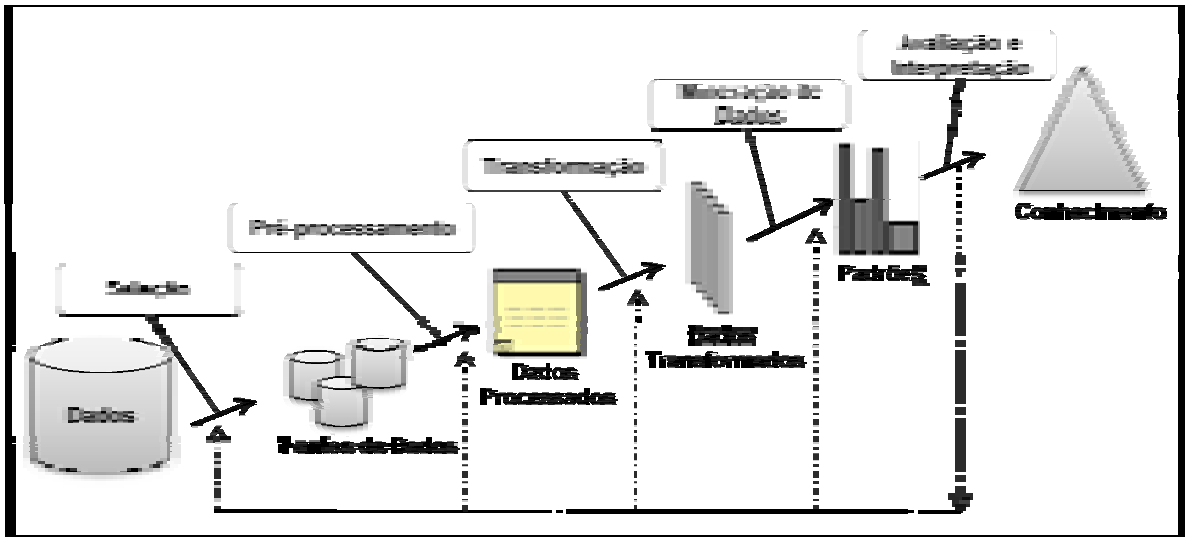


Figura 9 – Visão Geral do Processo de DCBD
Fonte: Adaptado de (Fayyad, Piatetsky-Shapiro e Smyth, 1996, p. 41)

Trata-se de um processo cíclico, podendo ser executado quantas vezes forem necessárias para a obtenção de conhecimento. As três primeiras fases dizem respeito à qualidade dos dados, sendo elas: **Seleção:** Captura do conjunto ou amostra de dados a serem manipulados; **Pré-processamento:** Limpeza dos dados, tais como ruídos, inconsistências,

dados inexistentes ou incompletos, que podem gerar padrões distorcidos; **Transformação:** Eliminação de atributos redundantes, padronização do conjunto de valores de domínio das variáveis. Estas fases convergem para a formação de uma base de dados concisa, em que dados de diferentes fontes e em diferentes formatos sejam coletados e armazenados em um único conjunto de dados. A próxima fase é a da **Mineração de Dados**, que é o foco deste trabalho, trata-se da aplicação de algoritmos para extração de padrões de relacionamento entre os dados. Finalizando o processo é aplicada a **Interpretação / Avaliação** dos resultados por especialistas no negócio, que avaliam os padrões identificados em função dos objetivos iniciais. Esta fase indica se o processo de DCBD deve ser repetido até a obtenção de resultados satisfatórios, ou se foi obtido conhecimento útil. Pode ser que o conhecimento obtido, do ponto de vista estatístico, esteja correto, porém pode não ser interessante ou de grande complexidade e difícil compreensão. Por isto, não basta que o processo esteja correto, mas que seja interessante do ponto de vista do negócio a ser explorado.

2.6 MINERAÇÃO DE DADOS (MD)

A Mineração de Dados é uma fase da DCBD, que busca identificar padrões de comportamento que se repetem. É nessa fase que efetivamente a busca pelo conhecimento é realizada. Algoritmos computacionais são aplicados na base de dados, em diferentes técnicas, com o auxílio de várias ferramentas ou *softwares*. A Mineração de Dados é também conhecida no termo em inglês como *Data Mining* (DM).

> Mineração de dados é definida como o processo de descoberta de padrões em dados. O processo pode ser automático ou (mais frequentemente) semiautomático. Aborda a resolução de problemas através da análise de dados já presentes em bancos de dados. Os padrões descobertos devem ser significativos, na medida em que levam a alguma vantagem, geralmente uma vantagem económica. (Witten e Frank, 2005, p. 5)

Os campos da Mineração de Dados envolvem a Estatística, a Inteligência Artificial e o aprendizado de máquina. Podem ser citadas como as principais tarefas: Agrupamento, Associação e Classificação, as quais envolvem métodos computacionais chamados algoritmos, que podem ser aplicados em diversas áreas de estudo, tais como: Redes Neurais Artificiais; Algoritmos Genéticos; Árvores de Decisão; Regras de Associação; Análise de Agrupamento; Regressão; Classificação Bayesiana; Análise Discriminante e Técnicas de Visualização.

Na fase de mineração de dados muitas técnicas tradicionais tem sido aplicadas com êxito. Pesquisas recentes demonstram que técnicas heurísticas tais como Algoritmos

Genéticos e Programação Genética têm sido utilizados com sucesso na tarefa de predição de regras de classificação. Neste estudo pretende-se investigar a aplicação destas heurísticas na tarefa de classificação.

2.7 TAREFA DE CLASSIFICAÇÃO

A tarefa de Classificação tem como objetivo encontrar um conjunto de modelos ou funções que descrevam e distingam classes de dados, de forma que esses modelos possam ser usados para predição. O modelo derivado é baseado na análise de um conjunto de dados de treinamento (conjunto de treino) e um conjunto de regras que podem ser usadas para classificar os padrões futuros. Este modelo é conhecido como classificador.

O conceito de Classe, segundo a Análise Orientada a Objetos, diz que:

> Classe é a representação de um conjunto de coisas reais ou abstratas que são conhecidas como sendo do mesmo tipo por compartilhar as mesmas características de atributos, operações, relações e semânticas. (Furlan, 1998, p. 16).

Um classificador é uma função que toma um conjunto ou lista de objetos como entrada e assume uma classe para ele. Assim, o classificador deverá identificar a qual classe este objeto pertence. Débora Carvalho em seu estudo "*Árvore de Decisão / Algoritmo Genético para Tratar o Problema de Pequenos Disjuntos em Classificação de Dados*", aborda os conceitos da tarefa de classificação da seguinte forma: (Carvalho, 2005, p. 7-8)

> Cada classe corresponde a um padrão único de valores dos atributos previsores. Esse padrão único pode ser considerado a descrição da classe. O conjunto de todas as classes é definido como C, e a cada classe Ci, correspondem uma descrição Di das propriedades selecionadas. Desta forma, usando estas descrições é possível construir um classificador o qual descreve um exemplo e do conjunto de exemplos T como sendo um exemplo pertencendo à classe Ci, quando aquele exemplo satisfaz Di.
>
> Um classificador extraído de um conjunto de dados serve a dois propósitos: predição de um valor e entender a relação existente entre os atributos previsores e a classe. Para cumprir o segundo propósito é exigido do classificador que ele não apenas classifique, mas também explicite o conhecimento extraído da base de dados de forma compreensível.
>
> A fim de contribuir para a compreensibilidade do conhecimento descoberto (relação entre os atributos e as classes), esse conhecimento é geralmente representado na forma de regras "se"..(condições).. "então"..(classe).., cuja interpretação é: "se" os valores dos atributos satisfazem as condições da regra "então" o exemplo pertence à classe prevista pela regra.

Um classificador pode ser representado através de árvores de decisão, algoritmos genéticos, redes neurais artificiais, redes bayesianas e fórmulas matemáticas, dentre outras.

A tarefa de classificação torna-se difícil quando o número de possíveis combinações de diferentes parâmetros é alto. Assim, uma heurística que explora soluções de grandes

dimensões, sem realizar buscas exaustivas e fornece regras de classificação inteligentes são muito atraentes. Neste sentido, a Programação Genética tem sido investigada para a descoberta de regras de classificação. Este estudo é citado por (Falco, Cioppa e Tarantino, 2002, p. 2).

2.7.1 TRABALHOS RELACIONADOS À APLICAÇÃO DE PG NA TAREFA DE CLASSIFICAÇÃO

A Programação Genética tem sido investigada como modelo de solução para a descoberta de regras de classificação. Uma população composta por um conjunto de regras candidatas é mantida e melhorada em um processo evolucionário, até que a qualidade da regra seja suficiente ou um critério de terminação seja atingido. O modelo permite a descoberta de regras que abrangem as diferentes classes.

Detalhes da aplicação da GP na tarefa de classificação serão descritos através de dois trabalhos que abordam a geração de regras de classificação. O primeiro é um dos exemplos citados por John Koza como resultados do experimento da Programação Genética aplicada em classificação de padrões, que pode ser visto em: (Koza, 1990, p. 51-52), (Koza, 1991, p. 3-5) e (Koza, 1992, p. 439-442). O segundo é um artigo de (Falco, Cioppa e Tarantino, 2002) que usa a Programação Genética para a descoberta de regras de classificação.

"Concept Formation and Decision Tree Induction Using the Genetic Programming Paradigm" (Koza, 1991)

John Koza apresenta uma solução para indução de descoberta de árvores de decisão, para classificação e reconhecimento de padrões, usando PG. Ele desenvolveu o algoritmo ID3 (Quinlan, 1986), partindo da semelhança entre os princípios do ID3 e o paradigma da PG.

No algoritmo ID3, o objetivo é particionar um universo de objetos dentro de classes. Cada objeto é descrito em termos de vários atributos. O sistema gera uma árvore de decisão que pode ser usada para classificar um novo objeto corretamente em uma classe usando os atributos do novo objeto. Os pontos externos (folhas) da árvore de decisão especificam o valor esperado do atributo categórico (classe). Esses nós folhas correspondem ao conjunto de terminais. Os pontos internos correspondem aos atributos não-categóricos e cada arco a um possível valor do atributo. Estes pontos internos correspondem ao conjunto de funções. O objeto é descrito pelo caminho formado do nó raiz ao nó folha. A Figura 10 apresenta a árvore de decisão gerada pelo algoritmo ID3 (Koza, 1991, p. 4).

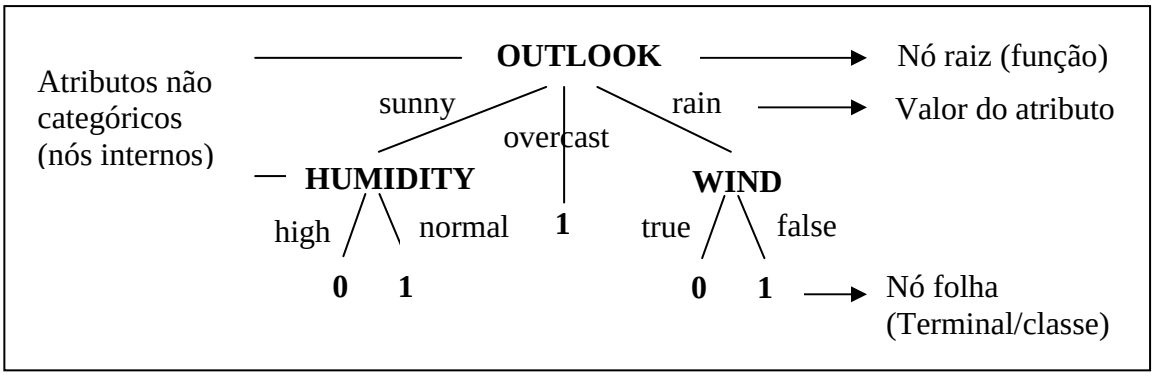


Figura 10 – Árvore de Decisão Gerada pelo Algoritmo ID3
Fonte: Adaptado de (KOZA, 1991, p.4)

Na Programação Genética indivíduos na população também são composições hierárquicas de funções e argumentos de vários tipos e formas, que são manipuladas por uma expressão simbólica escrita em uma linguagem de programação. As operações genéticas de reprodução e recombinação, através da avaliação de uma função de aptidão (*fitness)* são usadas no processo evolutivo de melhoria da solução. O processo começa pela comparação do atributo de treinamento com o atributo da raiz da árvore. O resultado determina qual ramo da árvore de decisão deverá ser seguido. O processo continua para cada ponto interno encontrado e finaliza quando leva a um nó terminal.

Quinlan (1986) apresentou um exemplo ilustrativo simples, composto de 14 casos de treinamento. Cada objeto é composto dos atributos não-categóricos chamados: TEMP, HUMI, OUT e WIND. Cada atributo podendo assumir vários valores. Ex: TEMP (hot, mild, cool), HUMI (high, normal); OUT (sunny, overcast, rainy) e WIND (true, false). O atributo categórico (classe) podendo assumir os valores (positivo – classe1 ou negativo – classe 0). A Figura 11 mostra o conjunto de dados de treinamento (Figura 11-a), o conjunto de indivíduos, composto por expressões simbólicas (Figura 11-b) e a árvore de decisão correspondente à expressão simbólica (Figura 11-c).

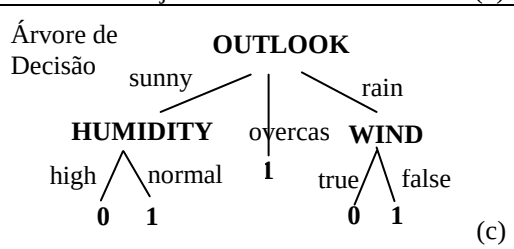
O processo de classificação usando PG ocorre da seguinte forma: Uma população inicial de indivíduos é gerada, conforme Figura 11-b. Esta população constitui o conjunto de regras de classificação. Cada regra é avaliada para cada um dos registros do conjunto de dados de treinamento (Figura 11-a), ou seja, todos os registros de treinamento são submetidos à mesma regra. O processo iterativo se repete até que todas as regras tenham sido avaliadas. O tipo de caminhamento na árvore de decisão durante a avaliação é do tipo pré-fixado. Um valor de *fitness* é calculado para cada regra.

OUT	TEMP	HUMI	WIND	PLAY
Sunny	Hot	High	False	0
Sunny	Hot	High	True	0
Overcast	Hot	High	False	1
Rainy	Mild	High	False	1
Rainy	Cool	Normal	False	1
Rainy	Cool	Normal	True	0
Overcast	Cool	Normal	True	1
Sunny	Mild	High	False	0

Conjunto de dados de treinamento (a)

(OUT (WIND (1 0) (WIND 1 1) (HUMI 0 1))
(HUMI (OUT 1 1 1) (WIND 1 0))
(HUMI (TEMP 1 1 1) (WIND 0 0))
(OUT (HUMI 1 0) (WIND 0 0) (WIND 1 0))
(HUMI (WIND 1 1) (WIND 1 0))

Conjunto de Indivíduos (b)



(c)

Figura 11 – Elementos da Solução para Indução de Descoberta de Árvores de Decisão, para Classificação e Reconhecimento de Padrões, usando PG
Fonte: Adaptado de (KOZA, 1991, p.3-4).

As regras são submetidas aos operadores de cruzamento e mutação, a fim de melhorar sua qualidade, sendo gerada uma nova população de indivíduos de regras. O processo é cíclico e repete até que a melhor regra seja encontrada, isto é, aquela que classifica corretamente todos os registros do conjunto de treinamento (Figura 11-a), ou que melhor se aproxime da solução ideal.

Os passos aplicados por John Koza para classificação do conjunto de treinamento, pela aplicação da PG, foram resumidos no Quadro 7 (Koza, 1992, p. 442).

Quadro 7 – Passos para indução à árvore de decisão

Item	**Descrição**
Objetivo:	Induzir uma árvore que classifica corretamente objetos Com atributos, dentro de classes.
Conjunto de classes ou terminais:	T={0,1}
Conjunto de atributos teste ou funções:	F={TEMP, HUMI, OUT,WIND}
Número de argumentos de cada atributo:	3,2,3,2
Fitness Cases*:*	14 casos de treinamento
Raw Fitness:	Número de treinamentos classificados corretamente
Standardized Fitness:	Fitness Cases – Raw Fitness
Parâmetros:	Tamanho da População: M=500 Quantidade de Gerações: G=51

Fonte: Adaptado de (Koza, 1992, p. 442)

Durante uma das execuções do algoritmo, a expressão do Quadro 8 emergiu na geração oito, classificando os 14 casos de treinamento corretamente:

Quadro 8 – Expressão ou Regra que Classifica os 14 Casos de Treinamento Corretamente

(OUT (WIND (1 0) (WIND1 1) (HUM 0 1)

"Discovering Interesting Classification Rules With Genetic Programming" (Falco, Cioppa e Tarantino, 2002)

Neste trabalho o objetivo é explorar a efetividade do método de PG para encontrar regras de classificação para problemas do mundo real. Os autores explicam que o objetivo é a construção de um sistema capaz de extrair automaticamente uma regra de classificação para cada classe em um banco de dados, a partir dos valores de alguns atributos, chamados atributos preditivos. Cada regra é constituída por uma combinação lógica desses atributos e essa combinação determina a descrição da classe. Uma vez definida a função de aptidão, o problema de classificação torna-se um problema de busca da melhor regra no espaço de busca de todas as soluções possíveis. O sistema permite obter regras que regem as diferentes classes através da execução do algoritmo evolutivo tantas vezes quanto o número de classes para predição. Quanto aos dados, estes são divididos em dois conjuntos: treinamento e testes. O conjunto de treinamento contém os objetos conhecidos e utilizados durante o processo de evolução para encontrar a regra de classificação explícita, enquanto o conjunto de testes é utilizado para avaliar a capacidade de generalização da regra encontrada. O algoritmo apresentado tem como base o método original de Koza (1992), sendo mostrado a seguir.

1. Gerar a população inicial de regras, representando soluções potenciais para o problema de classificação, para serem manuseadas;
2. Avaliar cada regra do conjunto de treinamento através de uma função de aptidão;
3. Selecionar as regras que serão submetidas ao mecanismo de reprodução;
4. Aplicar os operadores genéticos de seleção, cruzamento e mutação, para produzir novas regras;
5. Reinserir os descendentes para criar a nova população;
6. Repetir as etapas (3) à (5) até que uma regra de classificação aceitável seja encontrada ou o número máximo de gerações seja atingido.
7. Repetir os passos (2) à (6) até que uma regra seja definida para cada classe no banco de dados;
8. Atribuir a cada exemplo de treinamento e de testes, uma única classe. A importância do passo (8) é que em todas as etapas anteriores se lida com uma classe de cada vez. Como conseqüência deste fato, dado um banco de dados

> com classes “C”, pode acontecer que algumas amostras pertençam a mais de uma classe: elas são chamadas de casos indeterminados. Isso significa que elas são ou capturadas por mais de uma regra ou por qualquer regra.

Observa-se que a abordagem busca definir uma regra para cada classe do banco de dados, se preocupando com os casos de treinamento que podem ser classificados em mais de uma classe, chamados de casos indeterminados. Para corrigir o problema, os autores sugerem a realização de uma fase de pós-processamento para eliminar os casos indeterminados (Passo oito). Se o exemplo não atender a nenhuma regra, então é assumida a classe de fronteira de menor distância e se o exemplo atender mais de uma regra, então é assumido uma classe de fronteira que tenha a maior distância.

Os passos aplicados pelos autores para classificação do conjunto de treinamento, pela aplicação da PG, foram resumidos no Quadro 9. A aplicação foi testada em bases de dados disponíveis publicamente, sendo que seis exemplos foram executados sob as mesmas condições.

Quadro 9 – Passos Seguidos na Aplicação da PG segundo os autores Falco, Cioppa e Tarantino

Item	**Descrição**
Objetivo:	Apresentar uma ferramenta genérica de PG para encontrar regras de classificação.
Funções:	Operadores lógicos {AND, OR, NOT}; operadores relacionais {<.<=, >=, >}; operadores {IN, OUT}
Terminais:	Atributos ou valores no domínio dos atributos
População Inicial:	Indivíduos selecionados randomicamente dos conjuntos de Funções e Terminais. Porém, mesmo com a seleção randômica, durante a construção da árvore, algumas restrições são impostas: O nó raiz deve ser um operador lógico com alta probabilidade para AND e OR e baixa probabilidade para NOT. Os nós internos quando o pai for um operador lógico, seu primeiro filho é sempre um atributo A_i. O método *half-and-half* foi escolhido para criar as estruturas iniciais.
Função fitness:	A Função *fitness* assume um valor numérico indicando se um indivíduo d no espaço D está correto. A função pode ser explicada como a diferença entre o número atual de exemplos para o qual a regra de classificação pertence ou não pertence à classe e o número de exemplos para os quais existe alguma classificação incorreta, ou a condição de predição não é satisfeita. Existe ainda outro parâmetro incorporado na função *fitness* e está relacionado ao número de nós e a profundidade da árvore de regras codificadas.
Parâmetros:	M=2.000, G= 30, Probabilidade de Cruzamento = 0.8, Probabilidade de Reprodução=0.1, Probabilidade de Mutação=0.1

2.8 DESCRIÇÃO DO ESTUDO DE CASO

A Fundação de Amparo a Pesquisa do Estado de Goiás (FAPEG) é uma agência governamental de fomento à pesquisa científica, que concede os recursos captados para a pesquisa, através de um modelo de rede de colaboração científica, denominado “Redes Goianas de Pesquisa”. A base de dados formada pelas informações das Redes foi selecionada como estudo de caso para a aplicação do método de Programação Genética. A escolha ocorreu devido às características da composição das Redes, que permite a combinação de atributos dos grupos de pesquisadores, gerados pelos relacionamentos entre eles, dotando a aplicação de certa complexidade.

O modelo de Redes de Pesquisa adotado pela FAPEG é fundamentado na teoria dos pequenos grupos das Redes Sociais, o ambiente é dinâmico devido à facilidade para troca de informações e articulação entre os indivíduos, o que favorece a formação de novos grupos, tornando incremental o crescimento da Rede. A Análise de Redes Sociais é oriunda da Sociologia, Antropologia e da Psicologia Social. O estudo faz uso de modelagens matemáticas, especialmente da Teoria de Grafos da Ciência da Computação, para descrição do modelo.

> As redes apresentam algumas características comuns, tais como: objetivos compartilhados, construídos coletivamente; dinamismo e intencionalidade dos envolvidos; produção, reedição e circulação de informação; desconcentração do poder; multi-iniciativas; ambiente fértil para parcerias, oportunidade para relações multilaterais; configuração dinâmica e mutante. As relações são não-hierárquicas e, quando ocorre algum tipo de hierarquia, se dá com o intuito de facilitar a disseminação e o compartilhamento de informações. (Ribas e Ziviani, 2008, p. 5)

A FAPEG foi criada através da Lei nº 15.472, de 12 de dezembro de 2005 e tem como finalidade atuar no fomento às atividades de pesquisa científica, tecnológica e de inovação que possam contribuir para o desenvolvimento sócio-econômico e cultural do Estado de Goiás (GOIÁS, 2005).

A organização da FAPEG foi inicialmente fomentada por meio do Edital CT-Infra 008/2005 – MCT/FINEP/Ação Transversal – Projeto Estruturante dos Sistemas Estaduais de CT&I. Este projeto forneceu suporte à estruturação das Redes Goianas de Pesquisa coordenadas pela FAPEG.

2.8.1 REDES GOIANAS DE FOMENTO À PESQUISA

Em 2006, com base na Conferência Estadual de CT&I em Goiás (2005) e Conferência Nacional de CT&I (1986, 2001, 2005), foi promovido pela FAPEG, o evento

científico denominado Agenda Goiana de Programas de Fomento à Pesquisa, que reuniu pessoas dos setores de governo, empresarial e acadêmico-científico, com a finalidade de discutir e definir a composição das Redes Goianas de Pesquisa (RGP). Um dos resultados foi a definição das áreas transversais de atuação das Redes, denominadas de Áreas da Agenda Goiana, as quais são mostradas na Tabela 1.

Tabela 1 – Áreas da Agenda Goiana

Áreas da Agenda Goiana
1 – Qualidade de Vida
2 – Conhecimento e Expressão Humana
3 – Infra-estrutura e Processos Produtivos
4 – Desafios Estratégicos e Políticas Públicas
5 – Agronegócios, Desenvolvimento Rural e Fundiário
6 – Pesquisa Inicial e Fundamental

Fonte: (FAPEG, 2010)

A Resolução Normativa Nº 06/2007, do Conselho Superior da FAPEG, regulamenta o credenciamento das Redes Goianas de Pesquisa, caracterizando a Rede como a união de no mínimo três ou mais instituições, sediadas em Goiás, sendo pelo menos uma de educação superior, ou de pesquisa, com o objetivo de, em conjunto, viabilizar a execução de projetos de pesquisa e/ou desenvolvimento e contribuir significativamente para o desenvolvimento científico e tecnológico do Estado de Goiás (FAPEG, 2007).

O que difere as Redes Goianas de Pesquisa de outros conceitos é o seu caráter institucional. Sua organização é formada por instituições parceiras, que por sua vez apresentam seus pesquisadores representantes, infra-estrutura de apoio e recursos de contrapartida. Cada Rede possui um Coordenador, responsável pela articulação entre as instituições e membros da equipe. Os projetos de pesquisa são submetidos à seleção, via Edital de Chamada Pública, e conforme as regras do edital, pesquisadores Mestres ou Doutores apresentam seus projetos, através da Rede, sendo denominados líderes de projeto.

As instituições parceiras representam os mais variados tipos de segmentos da sociedade goiana: Governo Municipal, Estadual e Federal, nos poderes executivo, legislativo e judiciário; Indústria; Comércio; Organizações não governamentais; Instituições de ensino superior e instituições de pesquisa, dentre outras, localizadas nos diversos municípios do Estado. Ressalta-se que na constituição da Rede, deve haver pelo menos uma instituição de ensino superior ou de pesquisa.

A equipe da Rede é formada pelo coordenador e representantes das instituições parceiras. O coordenador é um pesquisador com titulação de Doutor e vínculo efetivo com

uma instituição de ensino superior ou de pesquisa. Os representantes das instituições parceiras são geralmente professores, alunos, técnicos e profissionais dedicados à pesquisa científica.

Em 2007 foi realizado o primeiro edital de chamada pública da FAPEG, a "CH01/2007: Fortalecimento de Redes de Pesquisa", um edital universal cujo objetivo foi o credenciamento das Redes Goianas de Pesquisa. O segundo edital, também universal, "CH02/2007: Programa de Fortalecimento da Ciência" veio apoiar a estruturação dos laboratórios ou núcleos de pesquisa parceiros das Redes de Pesquisa. Após estes editais o credenciamento das Redes se tornou fluxo contínuo e os fomentos distribuídos pela FAPEG: auxílio pesquisa e bolsa de formação foram vinculados ao fortalecimento das Redes.

A distribuição quantitativa de Redes por Área da Agenda Goiana, em 2010, é mostrada na Tabela 2.

Tabela 2 – Redes Goianas de Pesquisa por Áreas da Agenda Goiana

Redes de Pesquisa por Áreas da Agenda Goiana	Quantidade	%
1 – Qualidade de Vida	100	25,00%
2 – Conhecimento e Expressão Humana	55	13,75%
3 – Infra-estrutura e Processos Produtivos	36	9,00%
4 – Desafios Estratégicos e Políticas Públicas	70	17,50%
5 – Agronegócios, Desenvolvimento Rural e Fundiário	76	19,00%
6 – Pesquisa Inicial e Fundamental	63	15,75%
Total	400	100%

Fonte: (FAPEG, 2010)

O processo de colaboração na Rede de Pesquisa ocorre de duas formas: interna a uma determinada Rede e externa entre as Redes.

Na colaboração interna as relações são formais. Quando um projeto é submetido, o coordenador da Rede, o líder do projeto e a instituição de vínculo do líder do projeto são co-responsáveis por sua execução. As instituições parceiras fornecem os recursos de infra-estrutura necessários ao projeto e os membros da equipe do projeto são também membros da equipe da Rede de Pesquisa. A colaboração ocorre em relação aos projetos de pesquisa intrínsecos à Rede e estabelecidas no credenciamento da Rede ou na submissão dos projetos de pesquisa.

Na colaboração externa as relações são informais sem a interferência da FAPEG. O relacionamento entre instituições e pesquisadores é livre, de forma que uma instituição pode apoiar várias Redes e um pesquisador pode ser membro de várias equipes de projetos de

pesquisa, em diferentes Redes. A colaboração externa também pode ocorrer quando duas ou mais Redes se associam para formação de grupos de excelência com o objetivo de apresentação de projetos de alcance maior.

2.8.1.1 Topologia das Redes Goianas de Pesquisa

Paul Baran (1964), um dos precursores da Internet, trouxe o conceito de construção de uma Rede Distribuída que pudesse sobreviver a algum ataque do inimigo. Segundo ele, em uma Rede Centralizada (Figura 12-a), todos os "nós da rede" são conectados diretamente a um centro, que recebe dados de um "nó" individual e o encaminha ao seu destino. Se este centro ou o caminho entre os "nós" origem/destino são destruídos, a comunicação é cortada. A alternativa foi criar uma Rede Descentralizada (Figura 12-b) usando vários centros, porém cada "nó" individual ficando dependente de seu centro e dos caminhos para ele. A alternativa apresentada foi a criação da Rede Distribuída (Figura 12-c) que não teria nenhum centro, cada "nó" seria conectado ou entrelaçado a vários vizinhos, com vários caminhos possíveis para o envio de dados. Se um vizinho ou caminho fosse destruído, outro caminho estaria disponível. (Baran, 1964).

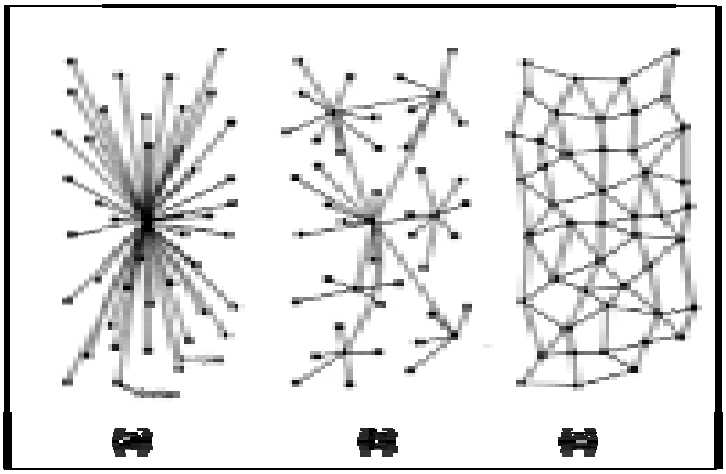


Figura 12 – Topologias de Rede, segundo Paul Baran
Fonte: Adaptado de (Baran, 1964)

Quanto à estrutura interna, a topologia da Rede Goiana de Pesquisa (RGP), é de uma rede centralizada, sendo o Coordenador, o mediador entre as instituições parceiras e os membros da equipe. Esta estrutura pode ser visualizada na Figura 13.

Quanto à estrutura externa, a topologia se caracteriza dentro do conceito de Rede Distribuída. Os relacionamentos entre cada "nó" se estabelecem pelos atores: instituição e pesquisador, que participam em outras Redes e acabam por compartilhar suas experiências. Dessa forma, a informação e o conhecimento circulam e novas parcerias podem ser formadas.

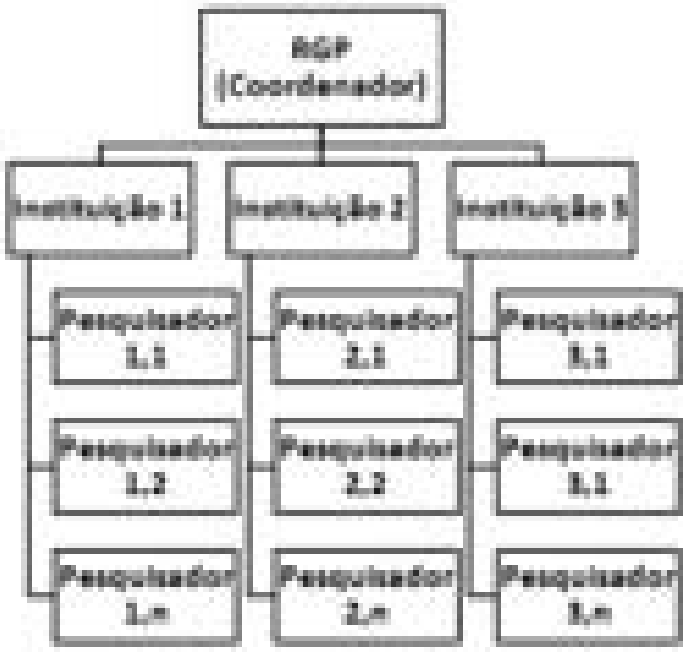


Figura 13 – Estrutura Interna Centralizada da Rede de Pesquisa

O coordenador age como facilitador da disseminação da informação, sem concentração de poder. A comunicação flui em várias rotas, pois um pesquisador pode ter vínculo efetivo com mais de uma instituição, podendo assim, ser membro de várias Redes, submetendo projetos diferentes em várias delas. Existem vários caminhos possíveis para o envio de projetos de pesquisa.

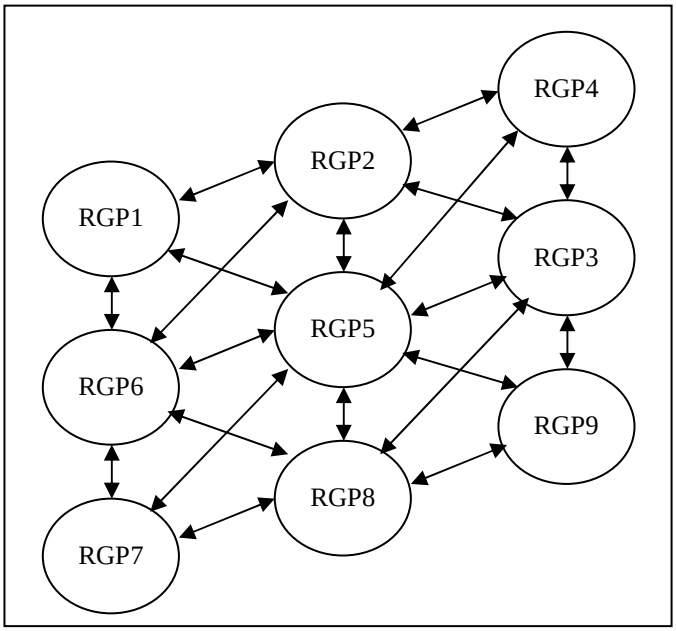


Figura 14 – Estrutura Externa Distribuída da Rede Goiana de Pesquisa (RGP)

A Figura 14 demonstra a possibilidade de ligações e combinações entre os “nós” das Redes Goianas de Pesquisa no modelo adotado pela FAPEG.

3 MATERIAL E MÉTODO

A metodologia usada neste trabalho envolve uma pesquisa bibliográfica em torno do tema Programação Genética, um algoritmo evolutivo que pode ser aplicado na fase de Mineração de Dados, uma das fases do processo de DCBD. Os principais instrumentos a serem utilizados serão: O método de PG proposto por Koza (1990) e o modelo de DCBD proposto por FAYYAD (1996). Ferramentas computacionais serão usadas como auxílio na aplicação dos métodos. A amostra de dados é composta por informações das Redes Goianas de Pesquisa da FAPEG.

3.1 PROBLEMA DA PESQUISA

O problema a ser explorado refere-se ao estudo da programação genética aplicada na classificação de processos de descoberta do conhecimento em bases de dados de redes de colaboração científica. A estratégia de Redes de Pesquisa usada pela FAPEG constituiu-se em uma iniciativa simples e inovadora, de estímulo e indução ao desenvolvimento científico no Estado de Goiás. O conceito de Redes Sociais aplicado na Gestão de CT&I torna-se um desafio por ser ainda um fenômeno recente. Além disso, tratar-se de um processo dinâmico que precisa ser continuamente estudado, com a identificação de indicadores e de novas lógicas de organização. O problema da gestão em Redes de Pesquisa configura-se em um problema complexo de decisão multi-critério, devido à combinação de atributos gerados pelos relacionamentos entre os grupos de pesquisadores. A combinação desses critérios leva à descoberta de Redes similares que poderiam ser induzidas a se unir para a formação de grupos consolidados, ou para o estabelecimento de eixos comuns em políticas públicas de CT&I.

3.2 HIPÓTESE

A aplicação do método de Programação Genética na tarefa de classificação poderá auxiliar na identificação de similaridades entre os grupos de pesquisadores que compõem as Redes de Colaboração Científica.

3.3 TESTE

O processo de DCBD é aplicado, com o auxílio de ferramentas computacionais, sendo testado em amostras da base de dados das Redes Goianas de Pesquisa, cedidas pela FAPEG para este trabalho. O algoritmo de Programação Genética é testado na fase de mineração sobre um conjunto de dados retirado dessa mesma base de dados. O volume de dados é de cerca de oito mil registros, com atualização em julho de 2010.

3.4 METODOLOGIAS APLICADAS

O estudo está limitado à aplicação do método de PG desenvolvido por (Koza, 1991), que usou um algoritmo evolucionário na indução à formação de árvores de decisão em processos de classificação de dados. O método serviu como base para o desenvolvimento de um algoritmo codificado sob a ferramenta MATLAB, o qual será aplicado sobre a base de dados das Redes Goianas de Pesquisa. Este método específico foi escolhido por ser Koza o autor da idéia original e principal fonte de referência.

Na preparação dos dados foi usado como metodologia o modelo de DCBD proposto por (Fayyad, Piatetsky-Shapiro e Smyth, 1996), composto pelas seguintes fases: seleção dos dados, pré-processamento, transformação, mineração de dados e interpretação. As ferramentas computacionais SGBD MYSQL, MATLAB e PENTAHO BI, conforme descrições a seguir, serão usadas como auxílio na aplicação do processo de DCBD e os resultados extraídos serão mostrados para ilustrar cada uma das fases. O algoritmo de PG é aplicado na fase de mineração de dados.

3.5 MATERIAIS USADOS

Quanto aos materiais bibliográficos as fontes de pesquisa foram: literaturas obtidas em livros, artigos em revistas científicas e publicações em bibliotecas digitais. Os materiais para o estudo de caso: SGBD, linguagem de programação e ferramenta computacional, são descritos a seguir:

3.5.1 SISTEMA GERENCIADOR DE BANCO DE DADOS (SGBD)

O SGBD usado para armazenar a base de dados das Redes de Pesquisa é o MySQL versão 1.2.10, fornecido pela empresa MySQL AB, o qual está instalado em um computador servidor, com sistema operacional Windows, na FAPEG.

3.5.2 LINGUAGEM DE PROGRAMAÇÃO

A linguagem de programação escolhida por John Koza para a definição da Programação Genética, segundo (Koza, 1992, p. 68) foi o LISP (LISt Processing). Porém, conforme o autor, virtualmente qualquer linguagem de programação (PASCAL, C, LISP) é capaz de expressar e avaliar a composição de funções e terminais para implementar a PG. Uma das razões da escolha do LISP é que funções e dados são tratados da mesma forma, sendo conhecida como uma linguagem funcional.

Neste trabalho será usada a ferramenta computacional MATLAB (*Matrix Laboratory),* desenvolvido pela companhia americana The Math Works, Inc. A ferramenta contém uma linguagem de programação de alto nível, destinada ao estudo de problemas científicos usando tabelas como a principal estrutura de dados. O MATLAB tem como vantagem o fato de possuir bibliotecas de funções pré-definidas, que reduzem o tempo gasto para realizar tarefas, além de ser também uma linguagem com paradigma funcional como o LISP.

3.5.3 FERRAMENTA PARA BUSINESS INTELIGENCE (BI)

Como auxílio às fases de preparação dos dados do processo de DCBD, será usado a ferramenta computacional denominada “Pentaho BI”. Trata-se de uma solução de código aberto, baseada em Java, desenvolvida pela comunidade de software livre mundial, sob a licença GNU – General Public Licence. A plataforma é centrada em processos, para criação de soluções para Business Intelligence (BI). É composta por vários módulos que podem trabalhar em conjunto ou individualmente. Inclui recursos de seleção de fontes de dados e integração; Data Warehousing (DW); Mineração de dados; Extração de dados; e Análise de informações, através da geração de relatórios e painéis para a criação de indicadores. (Pentaho, 2009).

A plataforma contempla todas as fases do Processo de DCBD, da seleção dos dados à análise de resultados, com cada aplicativo relacionado a uma dessas fases. Por se tratar de uma ferramenta com interface simples, interativa e visual, facilita principalmente o pré-processamento e avaliação dos resultados. Porém o aplicativo para a Mineração de Dados não

contempla o algoritmo de Programação Genética, que neste trabalho será substituído pela implementação na ferramenta MATLAB.

A ferramenta *Pentaho BI* pode ser obtida livremente pelo endereço *http://www.sourceforge.net/projects/pentaho*. Neste trabalho será usada a suíte de produtos Pentaho BI Server 3.0.0 – Stable, os módulos utilizados serão descritos a seguir, conforme (Pentaho, 2009).

Pentaho BI Server: Aplicativo que permite uma interface WEB padronizada e amigável, para a geração de Análises e Relatórios, pelos usuários. Conhecido como Pentaho User Console (PUC) – Interface com o usuário.

Pentaho Administration Console (PAC): Trata-se da interface para administração de usuários, permissões, configuração da base de dados, serviços e escalonamento de tarefas. Este aplicativo também possui interface WEB.

Pentaho Data Integration (PDI): Aplicativo que permite tratar as fases de seleção, pré-processamento e transformação dos dados, de forma visual, criando um passo-a-passo do que deve ser feito para que os dados sejam carregados corretamente na base de dados para mineração e exploração dos dados.

Pentaho Metadata Editor (PME): É o aplicativo que constrói o modelo de dados lógico do banco de dados, através do mapeamento do modelo físico. É usado junto com o PDI nas fases de seleção, pré-processamento e transformação dos dados.

Pentaho Report Desing (PRD): Aplicativo para a criação, geração e publicação dos relatórios através do Pentaho User Console (PUC). Usado na fase de interpretação dos dados, como auxílio na avaliação e análise dos resultados.

A Figura 15 mostra a relação entre cada fase do processo de DCBD e a ferramenta computacional aplicada.

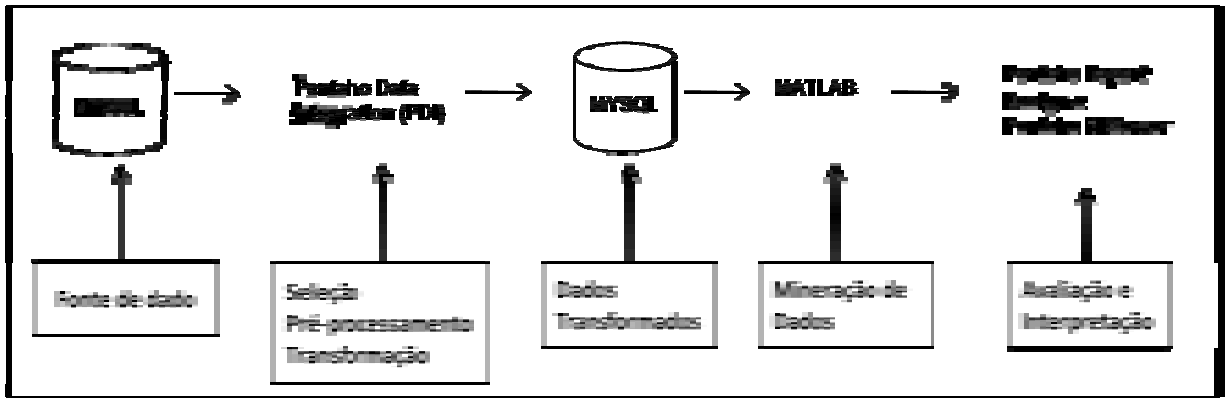


Figura 15 – Relação entre o Processo de DCBD e Ferramentas Computacionais

4 RESULTADOS ESPERADOS

Resultados foram sendo obtidos durante a aplicação de cada uma das fases do processo de DCBD, mostrado na Figura 16. As fases que antecederam a Mineração de Dados foram necessárias para preparar o conjunto de dados para aplicação do método da PG. De acordo com esse processo os resultados serão mostrados a cada fase, sendo detalhado para a PG que é o foco do estudo.

4.1 DCBD - FASE-1: FONTES DE DADOS

A origem dos dados foi identificada com base no diagrama de Macro-Processos das Fundações de Amparo a Pesquisa, apresentado pelo Projeto SIFAPs (2009) e adaptado para este estudo, conforme

Figura 16.

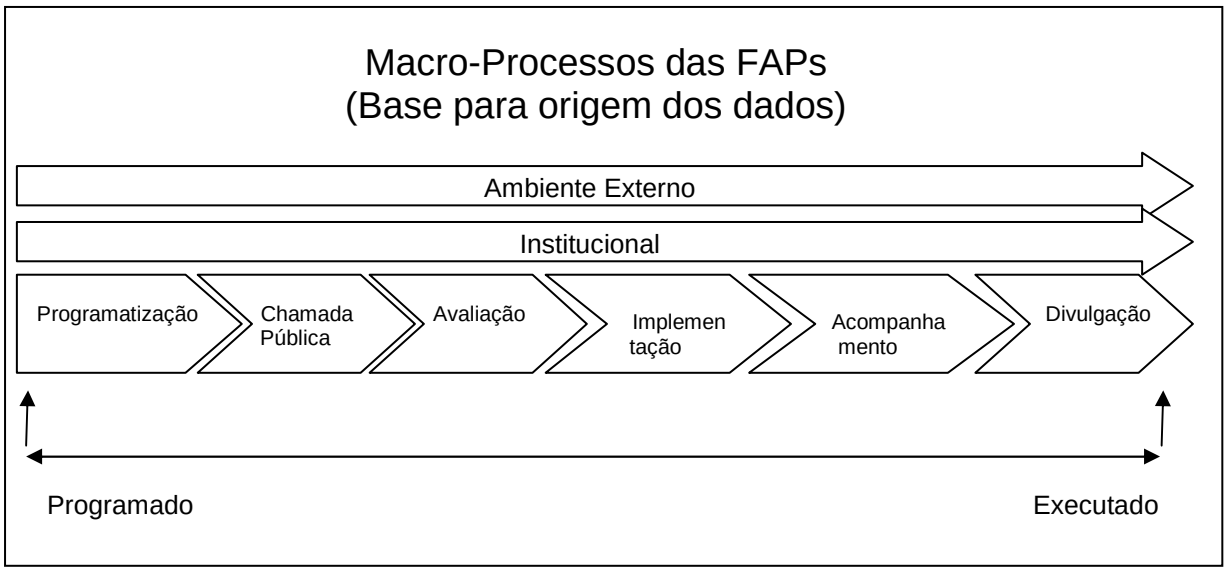


Figura 16 – Relação dos Macro-Processos da Fundação de Amparo a Pesquisa
Fonte: Adaptado de (SIFAPs, 2009)

Em cada macro-processo, informações são geradas e inseridas na base de dados através do sistema de informações da FAPEG, denominado FAPEGestor. A base de dados das Redes Goianas de Pesquisa faz parte deste sistema, sendo esta a principal fonte de dados.

4.2 DCBD - FASE-2: SELEÇÃO, PRÉ-PROCESSAMENTO E TRANSFORMAÇÃO

Atributos foram selecionados da fonte de dados identificada, sendo o resultado mostrado no Quadro 10. As tabelas referentes às propostas de projetos foram juntadas, através de comandos da linguagem de banco de dados SQL, resultando em uma única tabela denominada fapeg_propostas. Esta junção envolveu as tabelas: fapeg_proposta, fapeg_proposta_02, fapeg_proposta_03, fapeg_proposta_04, fapeg_proposta_05, fapeg_proposta_06, propostas, planos.

Quadro 10 – Relação de Atributos Resultantes da Base de Dados do Sistema FAPEGestor

Fonte de Dado	**Tabela**	**Atributo**	**Qtde Registros**
Sistema FAPEGestor	fapeg_areas_agenda_goiana	id_area descricao	6
	fapeg_areas_agenda_goiana_sub	id_sub_area id_area descricao	36
	fapeg_instituicao	id_instituicao instituicao sigla	411
	fapeg_rede	id_rede titulo coordenador área tema	400
	fapeg_instituicao_rede	id_instituicao id_rede data	1381
	fapeg_usuarios	id user nome cpf	747
	fapeg_propostas	controleDigital nr_Chamada ano_Chamada id_Rede tituloProjeto id_Lider situacao valor_Recomendado	666
	Editais	edit_iden edit_numr_chamada edit_ano_chamada edit_titulo	21

Um novo modelo conceitual foi concebido e o modelo físico criado no Sistema Gerenciador de Banco de Dados MY-SQL, em um computador local com Sistema Operacional Windows XP. A base de dados contém um volume de cerca de 8.000 registros, que foram atualizados em novembro de 2010.

Nesta fase foram utilizadas as ferramentas Pentaho Data Integration (PDI) e Pentaho Metadata Editor (PME). O objetivo foi obter um conjunto de dados transformados, integrados e concisos, para extração de dados via programação SQL e aplicação da Mineração de Dados. O modelo de dados resultante é mostrado na Figura 17 e descrito a seguir.

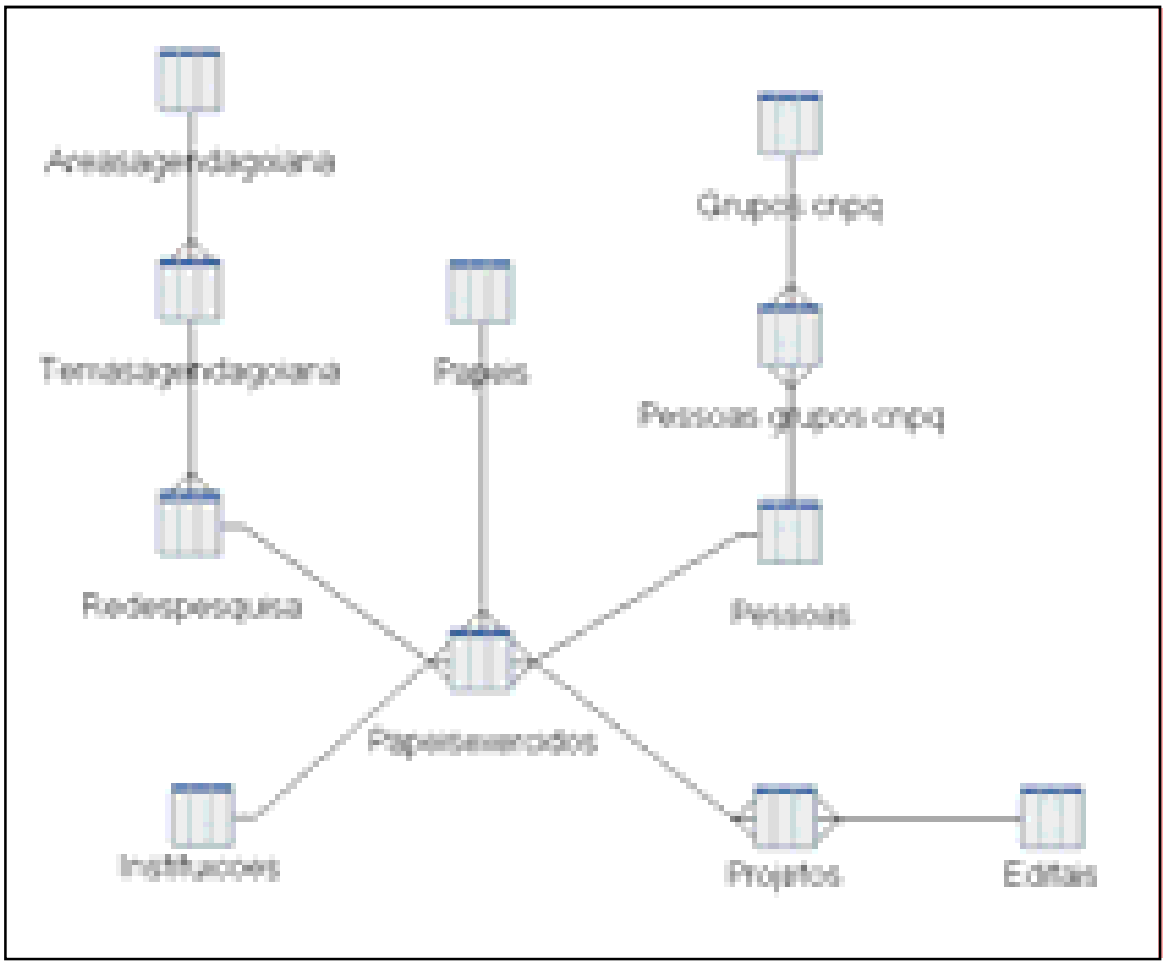


Figura 17 – Modelo de Dados Resultante das Fases de Pré-Processamento e Transformação

O núcleo principal do modelo de dados é formado pelas tabelas “Redespesquisa”, “Instituicoes”, “Pessoas”, “Projetos” e “Papeisexercidos”. Os grupos de pesquisa do CNPq foram acrescentados visando à integração com a Plataforma *Lattes*, através do relacionamento dos pesquisadores com os Grupos do CNPq. A tabela “Editais” contém informações sobre as chamadas públicas da FAPEG. As tabelas “Areasagendagoiana” e “Temasagendagoiana” contêm informações sobre a Agenda Goiana de Fomento à Pesquisa. A tabela “Papeis” é uma normalização do tipo de colaboração, visando melhorar a padronização dos dados.

As relações de colaboração entre Redes, Pessoas, Instituições e Projetos são realizadas através da tabela “Papeisexercidos”. O resultado dos tipos de colaboração identificados é mostrado no Quadro 11.

Quadro 11 – Resultado dos Tipos de Relação de Colaboração Identificados

Papel Exercido	Tabelas Relacionadas	Descrição da Relação de Colaboração
Coordenador de Rede	Redespesquisa Papeisexercidos Pessoas Instituicoes	O pesquisador coordena determinada Rede, representando uma instituição.
Líder de Projeto	Redespesquisa Papeisexercidos Pessoas Projetos Instituicoes	O pesquisador lidera projetos em determinada Rede, representando uma instituição.
Orientador em PPGSS	Redespesquisa Papeisexercidos Pessoas Projetos Instituicoes	O pesquisador orienta projetos em determinada Rede, representando o PPGSS de uma instituição.
Equipe do Projeto	Redespesquisa Papeisexercidos Pessoas Projetos Instituicoes	Pesquisadores membros de equipes de projetos, em determinada Rede, representam várias instituições.
Instituição parceira da Rede	Redespesquisa Papeisexercidos Instituicoes	Instituições apóiam diversas Redes de Pesquisa

A descrição dos atributos do modelo de dados gerado (Figura 17), também chamada de “Dicionário de Dados” é mostrada no Quadro 12, que apresenta a tabela do banco de dados, os atributos com descrição e a quantidade de registros da tabela.

Quadro 12 – Dicionário de Dados Resultante do Pré-Processamento e Transformação

Tabela	Atibuto	Descrição do Atributo	Quantidade Registros
Papeisexercidos	idPapelExercido idPapel idInstituicao idRedePesquisa idProjeto idPessoa DataInicio DataFim	Identificador da tabela papéis exercidos Tipo de papel exercido Identificador da Instituição Identificador da Rede de Pesquisa Identificador do Projeto Identificador da Pessoa Data de início da relação Data final da relação	3297
Redespesquisa	idRedePesquisa tituloRede idTemaAgendaGoiana	Identificador da Rede de Pesquisa Título da Rede Identificador do tema da agenda goiana	400
Instituições	idInstituicao nomeInstituicao siglaInstituicao	Identificador da Instituição Nome da instituição Sigla da instituição	424
Pessoas	idPessoa nomePessoa cpfPessoa	Identificador da Pessoa (Pesquisador) Nome do pesquisador CPF do pesquisador	1628

Projetos	idProjeto idEdital nomeProjeto nrProcesso valorRecomendado tipoSituacao tipoStatus TipoProjeto	Identificador do Projeto Identificador de Edital Título do Projeto Número do Processo Valor do projeto Identificador da situação do projeto Identificador do status do projeto Identificador do tipo de projeto	1501
Areasagendagoiana	idAreaAgendaGoiana nomeAreaAgendaGoiana	Identificador da tabela Descrição da área da agenda goiana	6
Temasagendagoiana	idTemaAgendaGoiana nomeTemaAgendaGoiana idAreaAgendaGoiana	Identificador da tabela Descrição do tema da área da agenda goiana Identificador da área da agenda goiana	36
Editais	id_edital numrEdital anoEdital tituloEdital valorEdital prazoEdital valorConvenioEst valorConvenioFed	Identificador da tabela Número da chamada do edital Ano da chamada do edital Título do edital Valor do edital Prazo do edital Valor do convênio estadual Valor do convênio Federal	30
Gruposcnpq	idGrupoCNPq nomeGrupoCNPq	Identificador da tabela Nome do grupo do CNPq	473
pessoasgruposcnpq	idPessoasGruposCNPq idGrupoCNPq idPessoa	Identificador da tabela Identificador do grupo do CNPq Identificador da Pessoa	691
papeis	idPapel nomePapel	Identificador da tabela Descrição do papel exercido na colaboração	5

4.3 DCBD - FASE-3: DADOS TRANSFORMADOS

Com os dados armazenados na nova base, foi possível descobrir alguns padrões de relacionamento estabelecidos com as instituições e pesquisadores, que podem auxiliar no fortalecimento das Redes de Pesquisa e conseqüentemente na tomada de decisão para a gestão da CT&I pela FAPEG. Estes resultados serão mostrados a seguir.

Nesta fase foram utilizadas as ferramentas Pentaho Report Design (PRD) e Pentaho User Console (PUC), além do SGBD MYSQL. Nos resultados obtidos serão mostrados apenas códigos, que representam os identificadores das tabelas, preservando a identidade dos pesquisadores.

4.3.1 RELACIONAMENTO DOS COORDENADORES COM DIFERENTES REDES DE PESQUISA

Com o objetivo de demonstrar este relacionamento será usado o exemplo em que coordenadores de Rede submetem propostas de projetos em outras Redes de Pesquisa, diferentes daquela coordenada por ele, como mostra a Tabela 3.

Tabela 3 – Coordenadores de Rede que são Líderes de Projetos em Outras Redes

Coordenador	Rede	Instituição	Coordenador (Líder)	Rede (Líder)	Instituição (Líder)
34	117	388	48	199	388
51	287	2	503	322	5
62	163	6	47	62	1
127	133	388	224	38	388
132	216	2	80	60	2
138	**105**	**1**	**263**	**19**	**40**
148	104	2	150	75	2
149	**101**	**2**	**131**	**79**	**2**
236	267	2	212	156	2
239	**132**	**5**	**138**	**105**	**1**
442	320	6	4	196	2
445	**174**	**2**	**149**	**101**	**2**
483	404	2	16	197	2
719	**392**	**2**	**131**	**79**	**2**
731	414	296	672	346	388
737	397	2	182	239	2

Trata-se de um relacionamento informal motivado por interesses comuns dos próprios pesquisadores. A Tabela 3 mostra que 16 (dezesseis) Coordenadores de Rede apresentaram projetos de pesquisa em outras Redes. Dois casos chamam a atenção: a) O relacionamento entre as Redes de número 19,105 e 132. O Coordenador da Rede 105 é líder de projeto na Rede 19 e o Coordenador da Rede 132 é líder de projeto na Rede 105. Observa-se ainda que os coordenadores são vinculados a diferentes instituições. b) O relacionamento entre as Redes 79, 101, 174 e 392. O Coordenador da Rede 101 apresentou projeto na Rede 79, o Coordenador da Rede 174 apresentou projeto na Rede 101 e o Coordenador da Rede 392 apresentou projeto na Rede 79. Neste caso, os coordenadores são da mesma instituição. Estes relacionamentos podem ser visualizados na Figura 18 e são indicativos de Redes em processo de fortalecimento, candidatas a participarem de um possível programa de indução à formação de grupos consolidados.

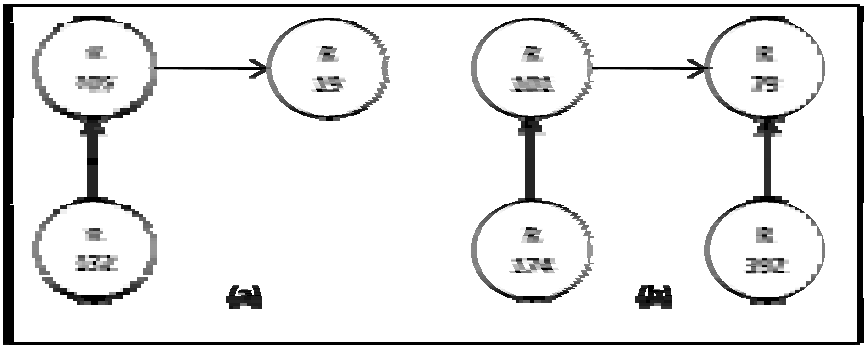


Figura 18 – Relacionamento entre as Redes através dos Coordenadores

4.3.2 RELACIONAMENTO DAS INSTITUIÇÕES COM AS REDES DE PESQUISA

Neste exemplo é possível observar o relacionamento entre as Redes de Pesquisa, através das instituições parceiras da Rede. Foram analisadas 371 Redes e 389 instituições.

A Tabela 4 apresenta a distribuição das instituições quanto às parcerias em Rede, considerando as 10 principais instituições em Rede. É interessante observar a significativa participação de instituições como as Secretarias Estaduais e Municipais de Saúde, e a Secretaria de Estado da Agricultura, Pecuária e Abastecimento, como parceiras de Instituições de Ensino Superior (IES), com objetivo de colaborar com o desenvolvimento científico em Goiás. Tais parcerias ocorrem no compartilhamento de laboratórios, equipamentos, ou mesmo, acesso a informações em bases de dados.

Tabela 4 – Distribuição das Instituições Parceiras em Rede

Instituição	Redes
Universidade Federal de Goiás (UFG)	273
Universidade Estadual de Goiás (UEG)	140
Pontifícia Universidade Católica de Goiás (PUC-GO)	139
Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA)	38
Universidade de Rio Verde (FESURV)	37
Instituto Federal Goiano - Campus Rio Verde (IF-RV)	36
Secretaria de Estado da Saúde (SES)	34
Secretaria de Estado da Agricultura, Pecuária e Abastecimento (SEAGRI)	33
Inst. Fed. Educação, Ciência e Tecnologia de Goiás-Campus Goiânia (IFG)	25
Secretaria Municipal de Saúde de Goiânia (SMS)	21

Observou-se ainda que 30% das instituições apóiam mais de duas Redes de Pesquisa, isto é, se relacionam com pelo menos duas Redes de Pesquisa, conforme Tabela 5.

Tabela 5 – Quantitativo de Instituições que Apóiam pelo menos duas Redes

Item	**Redes**	**%**
Instituições que Apóiam Duas Redes ou Mais	115	30%
Total de Instituições Parceiras em Redes	389	

As três principais instituições de ensino superior de Goiás, em quantidade de parcerias em Redes de Pesquisa da FAPEG, Universidade Federal de Goiás, Universidade Estadual de Goiás e Pontifícia Universidade Católica de Goiás, apóiam a maioria das Redes, porém em conjunto este apoio ocorre em somente 10% delas. Isto reforça a percepção de que o apoio das instituições é diversificado. Esta distribuição é mostrada na Tabela 6.

Tabela 6 – Redes Apoiadas Conjuntamente pelas Três Principais Instituições de Ensino Superior de Goiás, em quantidade de Parcerias em Redes de Pesquisa da FAPEG

Item	Redes	%
Redes apoiadas conjuntamente pela UFG, UEG e PUC-GO	39	10%
Total de Instituições	399	100%

4.3.3 RELACIONAMENTO DA EQUIPE DA REDE COM DIFERENTES REDES DE PESQUISA

Neste exemplo buscou-se investigar os pesquisadores líderes de projeto que mesmo não sendo coordenadores de Rede, estabelecem relacionamentos entre diferentes Redes de Pesquisa, conforme apresentado na Tabela 7. Foram identificados oito líderes de projetos que apresentaram projetos em pelo menos duas Redes de Pesquisa diferentes.

Tabela 7 – Coordenadores de Rede que são Líderes de Projeto em outras Redes

Líder de Projeto	Redes de Pesquisa
504	334, 322
528	316, 56
598	333, 345
600	60, 99
780	197, 119
10147	179, 301
10216	294, 407
10310	104, 122

A Figura 19 mostra o relacionamento entre as Redes 322 e 334, através do líder 504. Apesar dos dados em análise preliminar, percebe-se a evolução das Redes como alternativa para encaminhamento dos projetos de pesquisa por diferentes caminhos. O resultado demonstra Redes em processo de fortalecimento, com possíveis indícios de agrupamento.

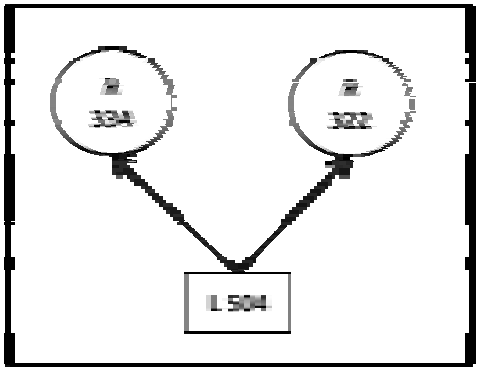


Figura 19 – Relacionamento entre as Redes através dos Líderes de Projeto

4.4 DCBD - FASE-4: MINERAÇÃO DE DADOS

4.4.1 DESCRIÇÃO DO PROBLEMA E SOLUÇÃO

O problema a ser resolvido na fase de mineração de dados refere-se à busca de Redes de Pesquisa similares, resultantes da combinação de atributos gerados pelos relacionamentos entre os grupos de pesquisadores. A similaridade entre duas ou mais Redes ocorre quando existe semelhança entre os valores de seus atributos.

Dado o conjunto de dados $X = \{x_1, x_2,..., x_n\}$, a semelhança entre x_1 e x_2 é dada pela freqüência dos valores dos atributos de x_1 em x_2. A Figura 20 mostra o conjunto de dados (X) e a Figura 21 demonstra a similaridade entre x_1 e x_5, onde os valores de três atributos de x_1 estão em x_5, sendo eles a "area", "equipe" e "instparc".

	area	equipe	instparc	grupocnpq	rede
x1	1	138	40	33	19
x2	1	263	1	228	19
x3	1	263	2	197	19
x4	1	263	245	197	19
x5	1	138	40	228	105
x6	1	239	286	393	105
x7	1	239	250	228	105

Figura 20 - Conjunto de Dados (X)

	area	equipe	instparc	grupocnpq	rede
x1	1	138	40	33	19
x5	1	138	40	228	105

Figura 21 - Similaridade entre Atributos do Conjunto de Dados (X)

A solução desenvolvida consiste de duas fases: a primeira diz respeito à construção do algoritmo de PG para descoberta de uma regra genérica de classificação. A segunda refere-se à construção do algoritmo classificador que usa a regra gerada para descobrir as Redes similares. Os algoritmos foram codificados sob a ferramenta MATLAB.

4.4.2 DESCRIÇÃO DO ALGORITMO DE PROGRAMAÇÃO GENÉTICA

O algoritmo de PG foi desenvolvido com base no método criado por John Koza (1990), que apresentou uma solução para indução de descoberta de árvores de decisão, para classificação e reconhecimento padrões. O método foi escolhido por ser Koza, o autor da idéia original e principal fonte de referência. Nesta fase o objetivo é a partir de um processo evolutivo, descobrir uma regra de classificação genérica que classifique corretamente os dados de treinamento.

Na segunda etapa, a melhor regra descoberta é aplicada sobre outro conjunto de dados de Redes de Pesquisa, contendo os mesmos atributos, porém, com uma amostra completa, a fim de predizer as classes similares. Esse passo constitui o "Classificador" propriamente dito.

Por analogia à biologia genética cada regra de classificação representa um cromossomo ou indivíduo da população. A estrutura de dados da regra corresponde à estrutura do cromossomo do indivíduo, com cada gene representado pela combinação lógica de atributos preditivos e valores do atributo classe, os quais determinam a descrição da classe. A Figura 22 mostra a estrutura de um cromossomo. A Figura 23 mostra a estrutura da regra de classificação, que é chamada por Koza (1992) de programa de computador.

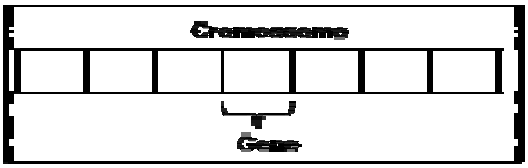


Figura 22 – Estrutura do Cromossomo

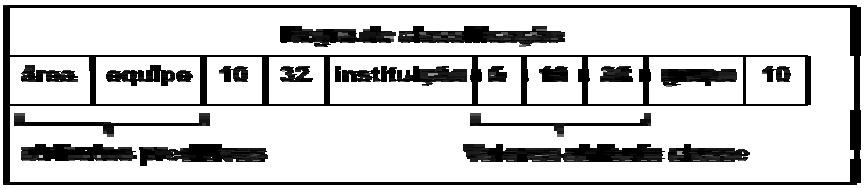


Figura 23 – Estrutura da Regra de Classificação

A regra de classificação permite várias combinações de condições "se... então... senão". Cada ramo da raiz à folha contempla uma situação, por exemplo, conforme a Figura 23 é possível cobrir as seguintes situações:

Se área = *argumento1* e equipe = *argumento1* então classe = 10
Se área = *argumento1* e equipe = *argumento2* então classe = 32
Se área = *argumento2* e instituição = *argumento1* então classe = 5
Se área = *argumento2* e instituição = *argumento2* então classe = 10
Se área = *argumento2* e instituição = *argumento3* então classe = 32
Se área = *argumento3* e grupo = *argumento1* então classe = 10

Cada nível da árvore avalia um atributo, sendo assim, quanto maior a quantidade de níveis, mais atributos serão avaliados. Uma árvore de quatro níveis permite avaliar até três atributos em conjunto. É preciso, porém cuidar para que na formação da árvore o atributo não se repita, neste caso pode-se realizar uma correção no ramo gerando um novo ramo a partir daquele ponto.

O Espaço de Busca para a formação da árvore é o conjunto de dados de treinamento $X = \{x_1, x_2,..., x_n\}$, composto por atributos das tabelas do banco de dados. De X são retirados

os valores de domínios dos atributos não-categórico da tabela do banco de dados (D); o conjunto de Aridades ou quantidade de valores de domínios do atributo (A) e o conjunto de terminais (T), onde cada elemento de T corresponde aos valores de domínio do atributo categórico ou classe.

A Propriedade de *Fechamento* é atendida porque cada elemento de F e T é retirado do próprio conjunto de valores de domínios das tabelas do banco de dados, fechando as combinações que possam ser encontradas. A Propriedade de *Suficiência* é atendida pela combinação de funções e terminais que irão formar a árvore de solução do problema, sendo suficientes para atender à solução.

Três critérios de terminação foram adotados: quantidade de gerações, melhor solução encontrada ou, solução aproximada encontrada pela convergência da regra.

Para avaliar a qualidade de um indivíduo, ou a aptidão da regra, são utilizadas as seguintes funções de *fitness: Raw Fitness (RF), Adjusted Fitness (AF) e Normalized Fitness (NF),* conforme fórmulas descritas na página 32 deste trabalho. O *RF* calcula o número de casos de treinamento classificados dentro das classes corretas. O *AF* ajusta o RF e varia entre zero (0) e um (1), se AF=1 indica que a melhor solução foi encontrada, isto é, a melhor regra que classifica todos os casos de treinamento corretamente. O NF é usado na seleção dos indivíduos pelo método Elitista.

O método “Elitista” foi usado para seleção dos indivíduos. A escolha deste processo é justamente para forçar uma convergência mais rápida do algoritmo. Os indivíduos correspondentes aos três melhores valores de “NF” são selecionados e copiados para a próxima geração sem modificações.

Do conjunto de melhores indivíduos são selecionados, dois indivíduos pais, para serem submetidos ao operador de cruzamento.

O operador de mutação é usado para melhorar a qualidade da regra, corrigindo a existência de ramos duplicados na árvore e ocorrência de elementos repetitivos no caminho do nó raiz ao nó folha.

A nova população é composta pelos indivíduos pais, selecionados pelo método elitista mais os indivíduos filhos gerados pelas operações de cruzamento e mutação.

Os parâmetros de controle iniciais são: quantidade de indivíduos (M), quantidade de gerações (G), profundidade máxima da árvore (pMax); valores dos atributos classe (classe1, classe2, classe3) e, número da execução do algoritmo.

Após a geração da população inicial foi acrescentada uma fase de avaliação da população, a fim de encontrar indivíduos ou regras duplicadas, de forma a melhor o desempenho do algoritmo. Essas regras são marcadas para que não sejam usadas durante o processo evolutivo.

O algoritmo construído a partir do algoritmo básico de Koza (1992), para descoberta da regra de classificação será descrito a seguir:

1. Inicializar os parâmetros de execução do algoritmo: conjunto de treinamento (X), conjunto de funções (F), conjunto de terminais (T), conjunto de domínios (D), conjunto de aridade (A), tamanho e profundidade da Árvore;
2. Gerar uma população inicial de composição randômica de funções e terminais do problema, formando a descrição da regra ou indivíduo;
3. Iterativamente realizar os passos 3.1 a 3.7 até que a quantidade de gerações seja atendida:
    3.1. Avaliar cada indivíduo da população e atribuir-lhe um valor de *fitness*, que expressa o quanto o quanto a regra está próximo da solução ideal;
    3.2. Verificar se a melhor regra foi encontrada. Se encontrar, mostrar solução e encerrar execução;
    3.3. Selecionar indivíduos para reprodução. Calcular o valor de *NF* e aplicar o método de seleção elitista;
    3.4. Verificar se a melhor regra aproximada foi encontrada pela convergência do algoritmo. Se encontrar, mostrar solução e encerrar execução;
    3.5. Aplicar o operador de cruzamento, pela recombinação genética de dois indivíduos pais, que são selecionados do conjunto de melhores indivíduos;
    3.6. Aplicar o operador de mutação para corrigir o problema de repetição de características no indivíduo, melhorando a qualidade de sua formação;
    3.7. Copiar os indivíduos gerados para a nova população.
4. Mostrar a solução encontrada.

Detalhamento dos Passos do Algoritmo Proposto

Passo1: Inicializar os parâmetros de execução do algoritmo: conjunto de treinamento (X), conjunto de funções (F), conjunto de terminais (T), conjunto de domínios (D), conjunto de aridade (A), tamanho e profundidade da Árvore;

a) *Conjunto de dados de treinamento (X):* Inicializado pelo acesso direto ao banco de dados, através do comando "*select"* da linguagem SQL.

b) *Conjunto de funções (F)*: nome dos atributos não-preditivos. Podem ser informados diretamente como parâmetro do algoritmo ou obtido através do comando "*mysql_list_fields"* da linguagem SQL.

c) Conjunto de terminais (T): valores do atributo classe.

$T = (X(1{:}\,n, m) - e)$

Onde:

X É o conjunto de dados de treinamento

n É o número de linhas de X

m É o número de colunas de X

e São os elementos repetidos em cada linha

d) Conjunto de domínios (D): valores dos atributos não-preditivos.

$D = (X(1{:}\,n, 1{:}\,m - 1) - e)$

Onde:

X É o conjunto de dados de treinamento

n É o número de linhas de X

m É o número de colunas de X

e São os elementos repetidos em cada linha

e) *Conjunto de aridades (A)*: quantidade de valores do atributo não-preditivo.

$A = tamanho(D(1{,}1{:}\,m))$

Onde:

D É o conjunto de domínios

m É o número de colunas de D

Passo2: Gerar uma população inicial de composição randômica de funções e terminais do problema, formando a descrição da regra ou indivíduo. A Figura 24 representa

a população inicial formada pelo conjunto de regras descritas através de expressões simbólicas, em notação pré-fixada, pelo método *Full* (Dolan, 2009).

(area (equipe 392 101 392) (instparc 101 101 101 79) (area 79 101 79))
(equipe (área 392 101 79) (grupocnpq 392 79 101 79) (area 392 101 79))
(area (equipe 392 101 392) (equipe 101 101 79) (instparc 79 392 79 79))
(area (grupocnpq 79 392 392 79101)(instparc 101 101 10179) (equipe 79 79 79))

Figura 24 – População Inicial Formada pelo Conjunto de Regras ou Indivíduos

Cada regra forma uma árvore hierárquica. A profundidade da árvore é informada como parâmetro inicial. O método *Full* (Dolan, 2009) foi usado por produzir árvores com ramos de mesma profundidade. A Figura 25 representa a árvore da regra: (equipe (instparc 392 79 101 79) (área 392 79 101) (grupocnpq 101 392 79)).

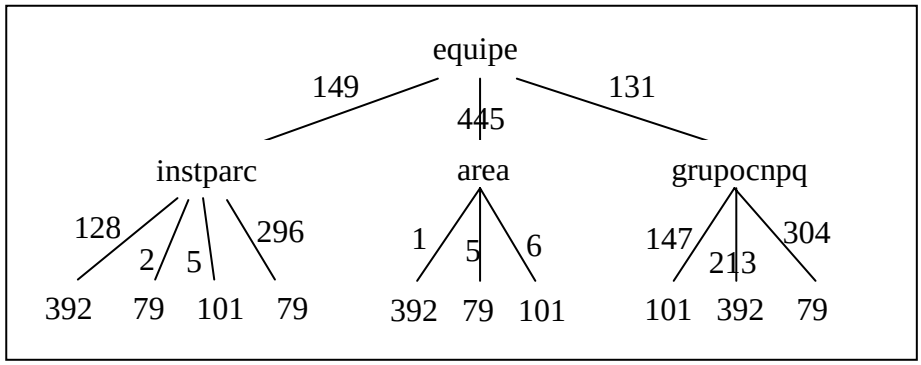


Figura 25 – Representação da Regra por meio de Árvore

Devido às limitações da ferramenta MATLAB em trabalhar com estruturas de dados em forma de árvores, foi codificada foi uma Matriz Dimensional (Y) de tamanho M de elementos numéricos.

$Y = Matriz(1:i, 1:j, 1:M)$, de números

Onde:

- i É o número de linhas da matriz
- j É o número de colunas da matriz
- M É o tamanho da população

Cada coluna da matriz contém informações do nó da árvore. A primeira coluna corresponde ao índice do elemento do vetor “F” ou do vetor “T”; A segunda coluna contém o indicador se o elemento é função ou terminal (1=função 2=terminal); A terceira coluna informa o nível do nó da árvore. A Figura 26 mostra a estrutura de dados do tipo “*Matriz*”, que representa os indivíduos da população.

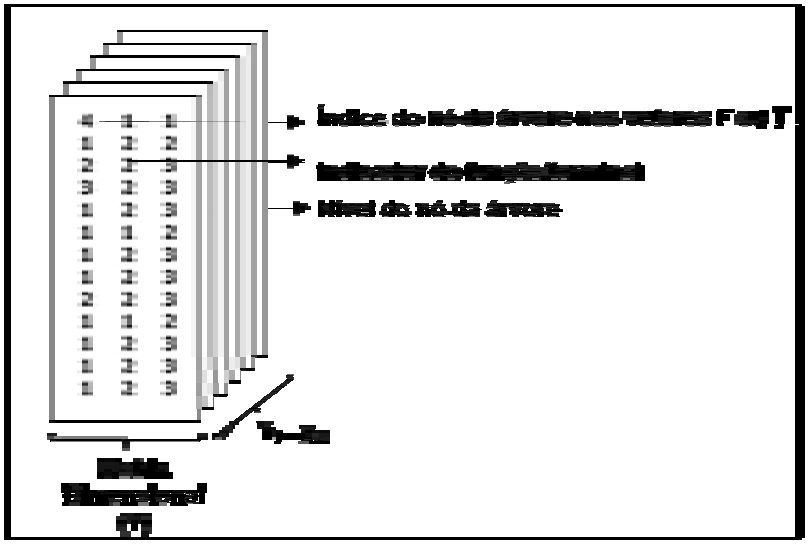


Figura 26 – Representação da Regra por meio da Matriz Dimensional (Y)

Após a geração da população inicial, esta é avaliada, a fim de encontrar regras duplicadas, melhorando o desempenho do algoritmo. Essas regras são marcadas para que não sejam usadas durante o processo evolutivo.

Passo 3.1: Avaliar cada indivíduo da população e atribuir-lhe um valor de fitness, que expressa o quanto a regra está próxima da solução ideal. Este passo envolve aplicar cada regra ($y_1...y_m$) sobre cada um dos elementos do conjunto de treinamento ($x_1...x_n$), a fim de calcular o valor de *fitness.*

Cada regra (y) corresponde a uma expressão simbólica em notação pré-fixada. Portanto é preciso realizar um caminhamento pré-fixado na árvore para sua avaliação.

O processo começa pela comparação de cada elemento de (x), com cada elemento da árvore de decisão (y).

Quando um elemento de x é apresentado à y, se o "nó da árvore" for uma função, os argumentos da função são testados a fim de determinar qual o ramo da árvore a ser seguido. O processo continua e finaliza quando leva a um nó terminal que corresponde ao valor da classe.

Cada elemento de x é avaliado, em cada nível da árvore y, se o nó é uma função, o valor do atributo é novamente avaliado, seguindo o caminho daquele ramo da árvore. Se o nó é um terminal, o valor do terminal é retornado em uma variável. Este valor é confrontado com o valor esperado para a classe de (x).

A Figura 27 mostra a aplicação de cada regra em cada elemento do conjunto de dados de treinamento.

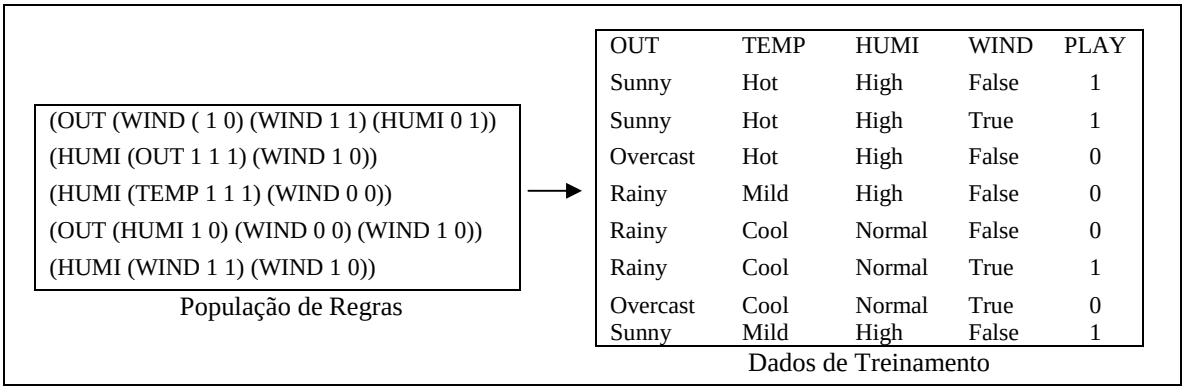


Figura 27 – Aplicação da Regra no Conjunto de Dados de Treinamento

Para cada linha da matriz X classificada corretamente é adicionado o valor um (1) ao *RF* daquela regra. A somatória de todos os valores de *RF* servirá de base para o cálculo *Adjusted Fitness (AF)* que indicará a qualidade da regra.

Passo 3.2: Verificar se a melhor regra foi encontrada. Se encontrar, mostrar solução e encerrar execução.

O *AF* ajusta o valor de RF e varia entre zero (0) e um (1), sendo que os maiores valores representam os melhores indivíduos. Portanto, se AF = 1 indica que a melhor solução, que classifica todos os casos de treinamento corretamente, foi encontrada e uma das condições de término foi atendida. O ciclo iterativo é finalizado e a melhor solução é apresentada. A fórmula para o cálculo do AF é mostrada na página 32 deste trabalho.

Passo 3.3: Selecionar indivíduos para reprodução. Calcular o valor de NF e aplicar o método de seleção elitista.

A seleção corresponde ao processo de reprodução assexuada, que seleciona um indivíduo conforme sua aptidão e o copia para a nova geração sem modificá-lo. No caso deste estudo foi usado o método Elitista justamente para forçar uma convergência mais rápida. O *NF* é calculado e os indivíduos correspondentes aos três melhores valores de *fitness* são selecionados e copiados para um conjunto de melhores indivíduos selecionados. A fórmula para o cálculo do NF é mostrada na página 32 deste trabalho.

Passo 3.4: Verificar se a melhor regra aproximada foi encontrada pela convergência do algoritmo. Se encontrar, mostrar solução e encerrar execução:

Neste passo é verificado se o tamanho do conjunto de melhores indivíduos selecionados é igual ao tamanho da população de indivíduos, se for igual, significa que o houve convergência do algoritmo e a solução a melhor solução aproximada foi encontrada. O ciclo iterativo é finalizado e a solução é apresentada.

Passo 3.5: Aplicar o operador de cruzamento, pela recombinação genética de dois indivíduos pais, que são selecionados do conjunto de melhores indivíduos:

Do conjunto de melhores indivíduos são selecionados dois indivíduos pais, para serem submetidos ao operador de cruzamento. Para cada indivíduo do conjunto de melhores indivíduos é selecionado outro par aleatoriamente. Um ponto aleatório de cruzamento é escolhido em cada indivíduo-pai e as árvores abaixo destes pontos são trocadas. A Figura 28 demonstra a aplicação do operador de cruzamento. O nível da árvore também é selecionado aleatoriamente. O algoritmo valida se os dois indivíduos selecionados são iguais e se a mesma posição de cruzamento foi escolhida, gerando uma nova posição, neste caso, para o segundo indivíduo.

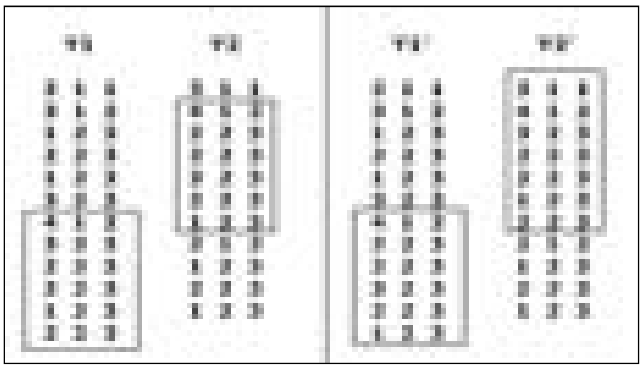


Figura 28 – Aplicação do Operador de Cruzamento

Passo 3.6: Aplicar o operador de mutação para corrigir o problema de repetição de características no indivíduo, melhorando a qualidade de sua formação:

O operador de mutação foi adotado visando melhoria da qualidade da regra. Dois tipos de correções são realizados: A primeira verifica se o nó raiz é repetido no caminho que vai da raiz à folha da árvore, se houver, um novo ramo é gerado a partir do ponto identificado. A segunda verifica se há ocorrência de ramos duplicados, sendo gerados novos ramos para os pontos identificados. A Figura 29 demonstra a operação de mutação.

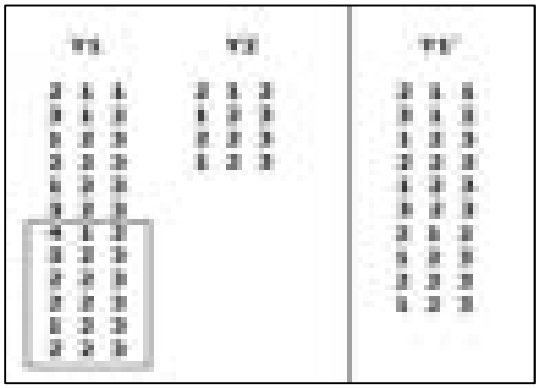


Figura 29 – Aplicação do Operador de Mutação

Passo 3.7: Copiar os indivíduos gerados para a nova população:

A nova população é composta dos indivíduos pais, selecionados pelo método elitista mais os indivíduos filhos gerados pelas operações de cruzamento e de mutação.

Passo4: Mostrar a solução encontrada:

Ao final da execução, o algoritmo apresenta como solução a regra que melhor se adapta aos dados do conjunto de treinamento. É mostrado o número da geração, número do indivíduo ou regra, a expressão simbólica, a quantidade de casos de treinamento, a quantidade e o percentual de casos classificados corretamente. O resultado mostra ainda a evolução da qualidade da regra de classificação, sendo avaliada pelo *Adjusted Fitness (AF)* em cada geração.

4.4.3 DESCRIÇÃO DO ALGORITMO CLASSIFICADOR

Na segunda etapa, a melhor regra encontrada é aplicada sobre o conjunto de dados de testes, formado pelos atributos preditivos e o atributo classe, conforme mostra a Figura 30. O objetivo do classificador é predizer quais seriam as outras classes, as quais o conjunto de atributos preditivos pertence, evidenciando a descoberta de novas relações.

Atributos Preditivos | Atributo Classe

area	equipe	instparc	grupocnpq	rede
1	138	40	33	19
1	263	1	228	19
1	263	2	197	19
1	263	245	197	19
1	138	40	228	105
1	239	1	393	105
1	138	286	228	105
1	239	250	393	132
1	239	2	393	132
1	239	1	197	132

Figura 30 – Representação do Conjunto de Testes

Os parâmetros de execução recebidos do algoritmo da PG são: melhor regra encontrada (y); conjunto de terminais (T); conjunto de valores de domínios dos atributos (D); profundidade máxima da árvore (pMax); valores do atributo classe (classe1, classe2 e classe3); número da execução do algoritmo (numrResult) .

A seguir será descrito em detalhes o algoritmo classificador construído a partir do algoritmo de PG desenvolvido neste estudo.

Algoritmo Classificador

1. *Inicializar o conjunto de dados de testes (X):*

   O conjunto de dados de testes é inicializado através de acesso direto ao banco de dados, por meio do comando *select* da linguagem SQL;

2. *Executar a classificação dos dados de teste (X):*

   Este passo envolve aplicar a regra de classificação (y) sobre cada um dos elementos do conjunto de testes $X=(x_1, x_{2,..}x_n)$. O processo é o mesmo realizado no Passo 3.1 do algoritmo da PG. Se os atributos preditivos se relacionarem com a regra (y), o valor da classe da regra será atribuído como uma nova classe de X.

   O algoritmo valida os casos em que nenhuma nova classe ou a mesma classe é encontrada, eliminando estes casos do resultado final.

   A Figura 31 demonstra o resultado do conjunto de testes após aplicação do classificador.

Atributos Preditivos				Atributos Classe	
area	**equipe**	**instparc**	**grupocnpq**	**rede**	**Nova classe**
1	**138**	**40**	**33**	**19**	**1**
1	**263**	**1**	**228**	**19**	**1**
1	**263**	**2**	**197**	**19**	**2**
1	**263**	**245**	**197**	**19**	**2**
1	**138**	**40**	**228**	**105**	**5**
1	**239**	**1**	**393**	**105**	**5**
1	**138**	**286**	**228**	**105**	**1**
1	**239**	**250**	**393**	**132**	**1**
1	**239**	**2**	**393**	**132**	**2**

Figura 31 – Representação do Resultado após Aplicação do Algoritmo Classificador

3. *Analisar a similaridade entre a classe original e a nova classe encontrada pelo classificador:*

   Considerando que os valores dos atributos preditivos da nova classe estão contidos no conjunto de valores de domínios (D), o objetivo será verificar se o valor de cada atributo preditivo de (X) pertence ao conjunto (D). A análise segue os seguintes passos:

   a) Mapear todos os valores dos atributos preditivos de (X) que pertencem a (D);

b) Verificar a freqüência dos atributos preditivos mapeados e criar uma nova coluna em (X) com a quantidade de atributos encontrados.

Quanto maior a freqüência de atributos encontrados, maior será a similaridade entre a classe original e a nova classe. A Figura 32 mostra a coluna “freqüência” que correspondem à quantidade de atributos preditivos mapeados.

Atributos Preditivos				Atributos Classe		
area	**equipe**	**instparc**	**grupocnpq**	**rede**	**Nova classe**	**Freqüência**
1	**138**	**40**	**33**	**19**	**1**	**3**
1	**263**	**1**	**228**	**19**	**1**	**3**
1	**263**	**2**	**197**	**19**	**2**	**3**
1	**263**	**245**	**197**	**19**	**2**	**4**
1	**138**	**40**	**228**	**105**	**5**	**4**
1	**239**	**1**	**393**	**105**	**5**	**3**
1	**138**	**286**	**228**	**105**	**1**	**3**
1	**239**	**250**	**393**	**132**	**1**	**3**

Figura 32 – Representação do Resultado após Análise da Similaridade

4. *Gravar resultado do classificador no banco de dados:*
   O resultado do classificador é inserido no banco de dados para aplicação da fase de avaliação e interpretação do processo de DCBD.

5. *Mostrar resultado do classificador:*
   O algoritmo gera um relatório das classes similares com pelo menos três atributos em comum, contendo: o conjunto de testes final; A descrição dos atributos comuns relacionados às classes; e a descrição das classes similares.

4.4.4 RESULTADOS OBTIDOS COM A APLICAÇÃO DO MÉTODO DE PG DESENVOLVIDO

A aplicação do método de PG será mostrada através de dois experimentos: O primeiro trata da reprodução do exemplo de (Koza, 1992, p. 440) sobre o conjunto de dados de treinamento de Quinlan (1986). O objetivo neste caso é demonstrar o funcionamento do método. Neste experimento é aplicada apenas a descoberta da regra usando PG. O segundo experimento é realizado sobre o conjunto de dados das Redes de Pesquisa, demonstrando a hipótese da viabilidade de se aplicar PG para classificação e identificação de relacionamentos e similaridades entre as Redes de Pesquisa.

Experimento I: Indução a árvore de decisão (Koza, 1992, p. 440)

O Quadro 13 apresenta um resumo dos critérios de execução para aplicação do método de PG desenvolvido neste trabalho, simulando o algoritmo ID3.

Quadro 13 – Experimento I: Resumo dos Parâmetros de Execução

Conjunto de classes ou terminais:	T= {0,1}
Conjunto de atributos teste ou funções:	F={TEMP, HUMI, OUT,WIND}
Número de argumentos de cada atributo (aridade):	A = {3, 2, 3, 2}
Conjunto de valores de domínios dos atributos preditivos:	D = {TEMP (hot, mild, cool), HUMI (high, normal); OUT(sunny, overcast, rainy) e WIND (true, false)}
Número de casos de treinamento (*Fitness Case):*	FC = 14 casos de treinamento
Parâmetros Iniciais:	Tamanho da População: M=200 Quantidade de Gerações: G=50 Profundidade máxima da árvore: pMax =3
Método de geração da população inicial:	Método *Full*
Operações genéticas:	Seleção, Cruzamento e Mutação
Método de seleção:	Seleção por Elitismo
Critérios de Parada:	Número máximo de gerações é atingido; Melhor solução é encontrada; O método converge para uma solução aproximada.
Estrutura de dados usada:	Matriz de células do tipo numérico
Resultado:	Melhor regra de classificação encontrada

Os dados de treinamento são mostrados no Quadro 14. Como a estrutura de dados usada é uma matriz numérica, os valores do conjunto de treinamento foram convertidos para o formato numérico, conforme coluna à direita do valor do atributo.

Quadro 14 – Experimento I: Conjunto de Dados de Treinamento

OUT		TEMP		HUMI		WIND		PLAY
sunny	1	hot	1	high	1	FALSE	2	0
sunny	1	hot	1	high	1	TRUE	1	0
overcast	2	hot	1	high	1	FALSE	2	1
rainy	3	mild	2	high	1	FALSE	2	1
rainy	3	cool	3	normal	2	FALSE	2	1
rainy	3	cool	3	normal	2	TRUE	1	0
overcast	2	cool	3	normal	2	TRUE	1	1
sunny	1	mild	2	high	1	FALSE	2	0
sunny	1	cool	3	normal	2	FALSE	2	1
rainy	3	mild	2	normal	2	FALSE	2	1
sunny	1	mild	2	normal	2	TRUE	1	1
overcast	2	mild	2	high	1	TRUE	1	1
overcast	2	hot	1	normal	2	FALSE	2	1
rainy	3	mild	2	high	1	TRUE	1	0

Após a execução do algoritmo proposto, a melhor regra de classificação emergiu na geração dois, com o indivíduo 10, classificando todos os casos de treinamento corretamente, conforme é apresentado no Quadro 15.

Quadro 15 – Experimento I: Resultado da Descoberta da Regra de Classificação

MELHOR SOLUÇÃO: Classifica todos os casos de treinamento corretamente	
Geração:	2
Individuo:	10
Expressão:	(OUT (HUMI 2 1) (WIND 1 1) (WIND 1 2)
Nr de casos de treinamento:	14
Nr de casos corretos (*rawfitness*):	14
Percentual de casos corretos:	100%

A Figura 33 apresenta a árvore de decisão da melhor regra de classificação encontrada na execução do Experimento I.

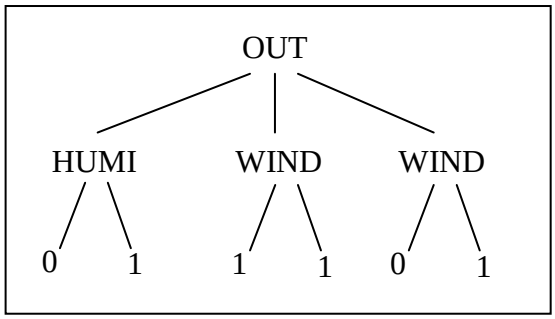


Figura 33 – Experimento I: Árvore de Decisão Correspondente à Melhor Solução

Experimento II: Dada uma determinada Rede de Pesquisa, o método classifica a base de dados, segundo a regra gerada, e identifica as Redes relacionadas, que indicam similaridade entre seus atributos.

O experimento escolhido para demonstração do método constitui-se no seguinte caso: “Dado um conjunto de três Redes de Pesquisa, deseja-se identificar as Redes relacionadas a elas, quanto aos atributos: Área da Agenda Goiana, Equipe, Instituições parceiras e Grupos de pesquisa do CNPq ao qual a equipe da Rede faz parte”. As Redes selecionadas para o experimento são: “79 - Rede Goiana de Pesquisa para Avaliação de Extratos de Espécies Vegetais do Bioma Cerrado quanto ao seu Potencial Citotóxico, Genotóxico e Farmacológico”; “101 - Rede Goiana de Pesquisa em Cultivo e Controle de Qualidade de Plantas Medicinais”; “392 - Rede Goiana de Pesquisa para Avaliação de Fatores Extrínsecos e Intrínsecos que Influenciam na Resposta Biológica a Fármacos e Correlatos”.

Os valores dos atributos foram codificados numericamente, a fim de facilitar a manipulação dos dados e melhorar o desempenho do algoritmo. O Quadro 16 apresenta um resumo dos critérios de execução para aplicação do método de PG, desenvolvido neste trabalho.

Quadro 16 – Experimento II: Resumo dos Parâmetros de Execução

Conjunto de classes ou terminais:	**T= {79, 101, 392}**
Conjunto de atributos teste ou funções:	**F= {area, equipe, instparc, grupocnpq, rede}**
Número de argumentos de cada atributo (aridade):	**A = {3, 5, 7, 12}**
Conjunto de valores de domínios dos atributos preditivos:	**D = {(area=1, 5,6); (equipe=131, 149, 445, 719, 1007); (instparc=2, 5, 128, 250, 296, 315, 378); (grupocnpq=5, 55, 116, 147, 157, 177, 213, 284, 304, 427, 443,460)}**
Número de casos de treinamento (*Fitness Case):*	**FC = 80 casos de treinamento**
Parâmetros Iniciais:	**Tamanho da População: M=150 Quantidade de Gerações: G=50 Profundidade máxima da árvore: pMax = 4**
Método de geração da população inicial:	**Método *Full***
Operações genéticas:	**Seleção, Cruzamento e Mutação**
Método de seleção:	**Elitismo**
Critérios de Parada:	**Número máximo de gerações é atingido; Melhor solução é encontrada; O método converge para uma solução aproximada.**
Estrutura de dados usada:	**Matriz de células do tipo numérico**

Os dados de treinamento foram selecionados da base de dados das Redes de Pesquisa da FAPEG, sendo os atributos preditivos: Área da Agenda Goiana (área); Equipe da Rede, considerando apenas o Coordenador e o Líder de Projeto (equipe); Grupo de Pesquisa do CNPq ao qual o Coordenado da Rede ou o Líder de Projeto é membro (grupocnpq); Instituições parceiras da Rede (instparc). O atributo classe corresponde ao Identificador das Redes de Pesquisa.

O teste realizado no Experimento II resultou em dez execuções, sendo a melhor solução, a regra número cinco com todos os casos de treinamento classificados corretamente e 14 Redes de Pesquisa com indicativos de similaridades. A Tabela 8 apresenta do resultado das execuções do teste.

Tabela 8 – Experimento II: Resultado do Teste da Execução do Algoritmo de PG Proposto

Resultado	Nr. Casos	Nr. Casos Corretos	% Casos Corretos	Geração	Indivíduo	Nr. Redes Similares
1	80	80	100%	39	30	04
2	80	76	95%	26	35	15
3	80	77	96.25%	50	7	04
4	80	80	100%	30	45	06
5	**80**	**80**	**100%**	**35**	**23**	**14**
6	80	62	77.5%	27	18	10
7	80	80	100%	49	5	06
8	80	80	100%	32	12	04
9	80	80	100%	39	7	09
10	80	77	96.25%	29	12	13

O resultado número cinco, considerado a melhor solução do teste, será detalhado a seguir. Neste caso, a melhor regra de classificação emergiu na geração 35, com o indivíduo 23, conforme é apresentado no Quadro 17.

Quadro 17 – Experimento II: Resultado da Descoberta da Regra de Classificação

MELHOR SOLUÇÃO: Classifica todos os casos de treinamento corretamente	
Geração:	35
Individuo:	23
Expressão:	(area (instparc (instparc 392 392 392 79 79 392 101) (instparc 79 392 392 79 101 101 392) (equipe 101 392 101 101 79) (instparc 392 101 79 79 392 101 101) (grupocnpq 79 79 101 101 79 392 79 392 101 392 79 101) (equipe 101 392 101 79 79) (instparc 79 392 392 79 101 101 392) (equipe (grupocnpq 79 101 101 79 101 392 101 79 392 392 101 79) (equipe 392 101 79 101 392) (equipe 392 101 101 101 79) (instparc 392 392 392 79 79 392 101) (grupocnpq 79 101 101 79 101 392 101 79 392 392 101 79) (instparc (instparc 79 392 392 79 101 101 392) (instparc 101 79 79 392 101 101 79) (instparc 101 79 79 392 101 101 79) (equipe 392 101 79 101 392) (instparc 392 392 392 79 79 392 101) (grupocnpq 79 392 79 79 101 79 392 79 79 79 101 101) (instparc 79 392 392 79 101 101 392)))
Nr de casos de treinamento:	80
Nr de casos corretos (*rawfitness*):	80
Percentual de casos corretos:	100%

A qualidade da regra de classificação é medida pelas funções *Raw Fitness e Adjusted Fitness.* O *Raw Fitness* calcula o número de indivíduos classificados dentro das classes corretas. O *Adjusted Fitness*, descrito pela fórmula (1), na página 32 deste trabalho varia entre zero (0) e um (1), sendo que os maiores valores representam os melhores indivíduos. A Figura 34(a) mostra o *Adjusted Fitness* em função do número de gerações. A Figura 34(b) mostra o *Raw Fitness* em função do número de gerações. O Quadro 18mostra a evolução da melhor regra de classificação relacionada à Figura 34.

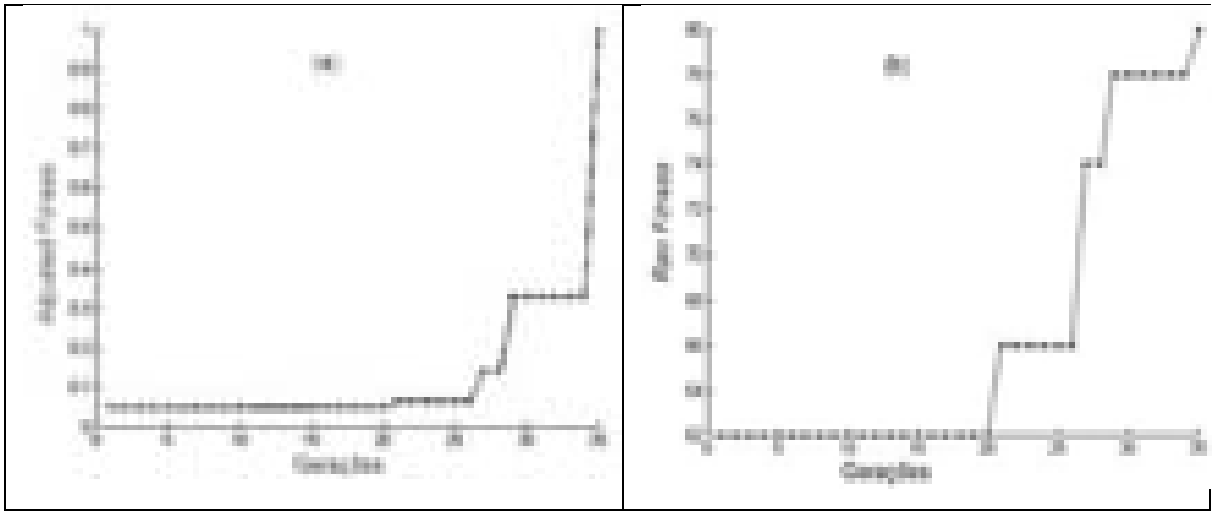


Figura 34 - Evolução da Melhor Regra de Classificação

Quadro 18 – Evolução da Melhor Regra Relacionada à Figura 34

Geração	Adjusted Fitness	Raw Fitness	Geração	Adjusted Fitness	Raw Fitness
1	0.0526	62	19	0.0526	62
2	0.0526	62	20	0.0526	62
3	0.0526	62	21	0.0667	66
4	0.0526	62	22	0.0667	66
5	0.0526	62	23	0.0667	66
6	0.0526	62	24	0.0667	66
7	0.0526	62	25	0.0667	66
8	0.0526	62	26	0.0667	66
9	0.0526	62	27	0.1429	74
10	0.0526	62	28	0.1429	74
11	0.0526	62	29	0.3333	78
12	0.0526	62	30	0.3333	78
13	0.0526	62	31	0.3333	78
14	0.0526	62	32	0.3333	78
15	0.0526	62	33	0.3333	78
16	0.0526	62	34	0.3333	78
17	0.0526	62	35	1.0000	80
18	0.0526	62			

Na segunda fase da solução, a melhor regra de classificação encontrada é então, aplicada ao conjunto de dados de testes, com 4.178 registros. O resultado final, após mapeamento dos atributos comuns, retornou 34 registros com pelo menos três atributos relacionados conjuntamente com as Redes 79, 101 e 392. Este resultado pode ser visualizado na Figura 35.

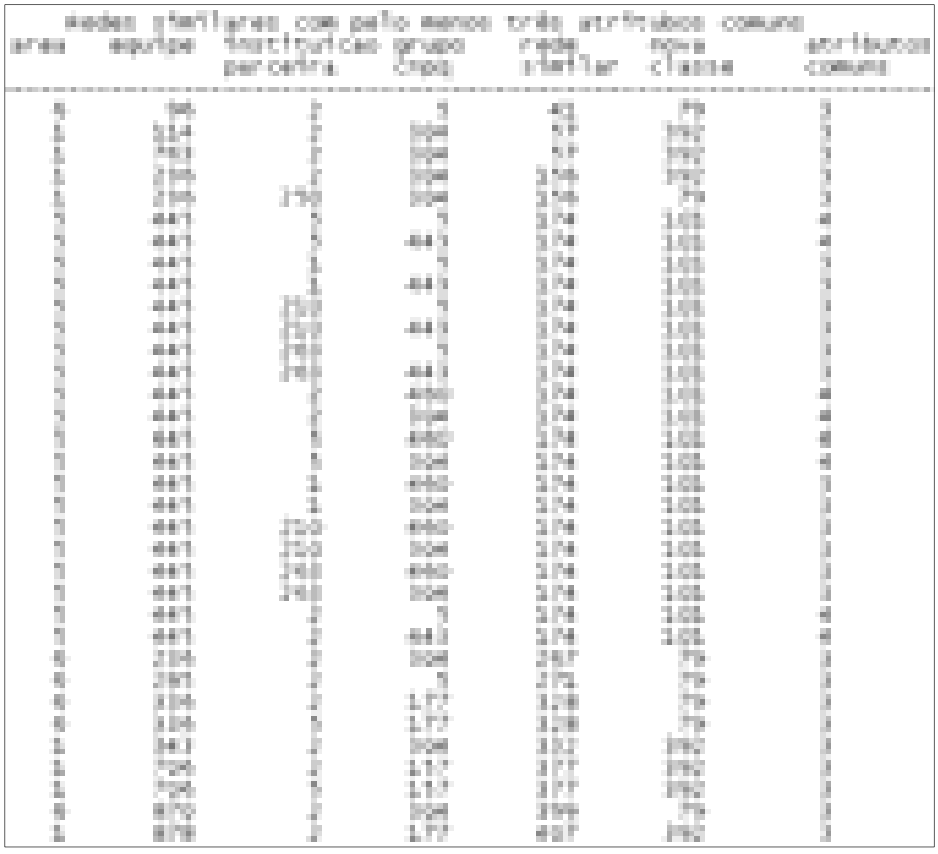


Figura 35 – Experimento II: Redes Similares Três ou mais Atributos Comuns

Neste resultado foram encontradas 14 Redes de Pesquisa com indicativos de similaridade, ou seja, que se relacionam às Redes 79, 101 e 392 em: Duas Áreas da Agenda Goiana, Seis Instituições Parceiras e Seis Grupos de Pesquisa do CNPq. A Tabela 9 mostra a descrição das Áreas da Agenda Goiana comum às Redes similares encontradas.

Tabela 9 – Experimento II: Descrição das Áreas da Agenda Goiana comum às Redes Similares

Nr	Descrição
1	Qualidade de Vida
5	Agronegócios, Desenvolvimento Rural e Fundiário

A Tabela 10 mostra a descrição das Instituições Parceiras da Rede de Pesquisa comum às Redes similares encontradas.

Tabela 10 – Experimento II: Descrição das Instituições Parceiras comum às Redes de Pesquisa Similares

Nr	Descrição
2	Universidade Federal de Goiás
250	Secretaria de Estado da Saúde
5	Universidade Estadual de Goiás
1	Pontifícia Universidade Católica de Goiás
210	Ledal Quimica do Brasil
263	Secretaria Municipal de Saúde de Goiania

A Tabela 11 mostra a descrição dos Grupos de Pesquisa do CNPq comum às Redes similares encontradas.

Tabela 11 – Experimento II: Descrição dos Grupos de Pesquisa do CNPq comum às Redes Similares

Nr	Descrição
5	Análise Farmacêutica e Ambiental - UFG
304	Núcleo de Estudos e Pesquisas Tóxico-Farmacológicas - NEPET-UFG
443	Grupo de Pesquisa em Plantas Medicinais e Fitoterápicos – UFG
460	Pesquisa, desenvolvimento e inovação tecnológica de produtos farmacêuticos, cosméticos e nutracêuticos a partir de matérias primas de origem natural, sintética ou biotecnológica - UFG
177	Grupo de Pesquisa em Bioatividade Molecular – UFG
157	Grupo de Estudo e Pesquisa em Endocrinologia, Metabolismo e Nutrição e suas Implicações na Saúde – UFG

As Redes Goianas de Pesquisa com indícios de similaridade, descobertas através da aplicação do método de Programação Genética (PG) estão descritas na Tabela 12.

Tabela 12 – Experimento II: Redes Goianas de Pesquisa Similares

Rede	Título da Rede
41	Rede Goiana de Pesquisa em Biossensores para Monitoramento Ambiental
57	Rede Goiana de Pesquisa em Monitorização Terapêutica, Toxicológica e Controle de Qualidade de Medicamentos
79	Rede Goiana de Pesquisa para Avaliação de Extratos de Espécies Vegetais do Bioma Cerrado quanto ao seu Potencial Citotóxico, Genotóxico e Farmacológico
101	Rede Goiana de Pesquisa em Cultivo e Controle de Qualidade de Plantas Medicinais
156	Rede Goiana de Pesquisa Vigilância à Saúde do Idoso no Estado de Goiás
174	Rede Goiana de Pesquisa, Desenvolvimento e Inovação de Produtos Cosméticos e Farmacêuticos a Partir de Plantas Medicinais
267	Rede Goiana de Pesquisa em Imunopatologia da Doença de Alzheimer
271	Rede Goiana de Pesquisa em Biodiversidade Microbiana em Cerrado Goiano
328	Rede Goiana de Pesquisa em Ressonância Magnética Nuclear
332	Rede Goiana de Pesquisa em Toxicologia Pre-Clinica e Clinica
377	Rede Goiana de Pesquisa em Doenças Cardiovasculares e Renais
392	Goiana de Pesquisa para Avaliação de Fatores Extrínsecos e Intrínsecos que Influenciam na Resposta Biológica a Fármacos e Correlatos
399	Rede Goiana de Pesquisa em Biofarmácia e Farmacocinética
407	Rede Goiana de Pesquisa em Epidemiologia e Caracterização de Agentes Etiológicos Causadores de Infecções Fúngicas Nosocomiais

4.5 DCBD - FASE-5: AVALIAÇÃO E INTERPRETAÇÃO

Os algoritmos desenvolvidos neste trabalho geram, ao final da execução, um relatório demonstrando o processo de descoberta da regra de classificação e o processo de classificação do conjunto de dados das Redes de Pesquisa, o qual é apresentado nos anexos. Os resultados foram também gravados no banco de dados MYSQL e precisam ainda passar pela interpretação de especialistas no negócio, a fim de avaliar os padrões identificados.

As Redes de Pesquisa escolhidas para o Experimento II foram selecionadas a partir do caso apresentado no item 4.3.1- Relacionamento dos Coordenadores com Diferentes Redes de Pesquisa, realizado na Fase 3 – Dados Transformados, do processo de DCBD, páginas 58 e 59 deste trabalho. Através de uma consulta ao banco de dados usando a linguagem SQL foi possível identificar o relacionamento entre as Redes 79, 101, 174 e 392. Porém a aplicação do algoritmo de PG e do algoritmo classificador, na Fase 4 – Mineração de Dados permitiu aprofundar a investigação sobre este caso preliminar, revelando o relacionamento dessas quatro Redes com outras 10, em um total de 14 Redes de Pesquisa com indicativos de similaridades.

O resultado do Experimento II identificou padrões de Redes de Pesquisa similares, conforme descrição do problema – página 61 deste trabalho, quanto aos atributos: Área da Agenda Goiana, Equipe, Instituições parceiras e Grupos de pesquisa do CNPq. Quatro atributos foram analisados, sendo encontradas 14 Redes com pelo menos três atributos relacionados conjuntamente.

Observa-se no resultado que dessas 14 Redes, nove possuem em seus títulos, palavras-chave relacionadas a medicamentos, plantas medicinais e produtos farmacêuticos, indicando estudos relacionados à área de fármacos. As outras cinco Redes, embora no título não se verifique relacionamento com estudos em fármacos, verificou-se que a equipe da Rede pertence a Grupos do CNPq[1] que trabalham nesta área. Portanto as 14 Redes poderiam ser agrupadas e fortalecidas em um programa de fomento específico ao estudo de fármacos em Goiás.

[1] Grupos do CNPq: Grupos de Pesquisa do Brasil constantes da base de dados do CNPq.

5 CONCLUSÃO E TRABALHOS FUTUROS

A teoria de Charles Darwin (1859) parte do princípio que indivíduos na população, geneticamente mais fortes ou qualificados, possuem maior probabilidade de chegar à fase adulta e gerar descendentes com as características genéticas transmitidas, aumentando essas características na população.

O método de Programação Genética, criado por Koza (1990), segue o mesmo princípio da teoria de Darwin e, fazendo uso dos conceitos da biologia genética, estrutura programas de computador como se fossem cromossomos. Nesse contexto, programas de computador viram regras de classificação compostas pela combinação de atributos de um objeto. A melhoria da regra passa por cruzamentos e mutações, sendo cada uma avaliada e as melhores selecionadas para a próxima geração, até que a regra "geneticamente mais forte ou qualificada" apareça como solução genérica para a classificação do conjunto de objetos.

Este trabalho investigou o potencial do método de programação genética como algoritmo classificador e com base no algoritmo proposto por Koza (1990), foi construído uma aplicação para geração de regras de classificação que pudessem ser aplicadas para descoberta de similaridades a partir do relacionamento entre grupos de pesquisadores de uma agência de fomento a pesquisa.

Durante o estudo pôde-se perceber o potencial do método na realização de combinações de critérios e condições lógicas. A possibilidade de mensuração da qualidade da regra a cada ciclo, conforme descrito na página 76 deste trabalho é uma das vantagens do método, destacando-se ainda, a simplicidade do algoritmo de PG.

Na construção do algoritmo de PG proposto, a dificuldade encontrada ocorreu em relação à possibilidade de geração de ramos duplicados na árvore de decisão correspondente à regra, ou mesmo de elementos repetitivos, os quais afetavam o desempenho do algoritmo e a qualidade dessa regra. Estes casos foram minimizados com a adoção do operador de mutação para correção do ramo da árvore e conseqüente aumento da diversificação dos elementos da regra.

A solução encontrada durante o processo de descoberta da regra de classificação, quando aplicada sobre a base de dados das Redes Goianas de Pesquisa vem auxiliar não

apenas na identificação de similaridades entre os grupos de pesquisadores, mas permite a descoberta de tendências de comportamento desses grupos. A ferramenta construída contribui para a gestão de agências de fomento que venham a adotar o modelo proposto pela FAPEG, por meio de estudos prospectivos para planejamento de políticas públicas de Ciência, Tecnologia e Inovação (CT&I).

Como estudo futuro sugere-se aprofundar a investigação sobre técnicas de melhorias da regra de classificação, desempenho do algoritmo de PG e significância estatística do processo, bem como a utilização do método proposto em demais estudos de caso em pesquisa operacional e simulação envolvendo o conceito de redes e grafos. Como ferramenta de gestão em agências de fomento à pesquisa, sugere-se ainda a investigação do método para uso na fase de julgamento e avaliação de propostas de projetos, visando a redução da subjetividade das avaliações.

REFERÊNCIAS

BARAN, P. On distributed communications: I. introduction to distributed communications networks. In: Memorandum RM-3420-PR, August 1964. Santa Mônica: The Rand Corporation, 1964. Disponivel em: <http://www.rand.org/pubs/research_memoranda/RM3420/index.html>. Acesso em: 26 jul. 2010.

CARVALHO, D. R. Árvore de Decisão / Algoritmo Genético para Tratar o Problema de Pequenos Disjuntos em Classificação de Dados. Tese de Doutorado, Universidade Federal do Rio de Janeiro - UFRJ, Rio de Janeiro - RJ. 162 p., 2005. Disponivel em: <http://www.ipardes.gov.br/webisis.docs/tese_deborah_carvalho.pdf>. Acesso em: 15 fev. 2010.

CASTRO, R. E. Otimização de Estruturas com Multi-objetivos Via Algoritmos Genéticos de Pareto. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro-RJ, p.202, 2001. Disponivel em: <www.lania.mx/~ccoello/EMOO/thesis_castro.pdf.gz>. Acesso em: 14 mai. 2009.

DANTAS, M. J. P. Metodologia de Síntese de Filtros de Microondas de Topologias Arbitrárias Utilizando Algoritmo Evolucionário Híbrido (Memético) associado a Conhecimento Especialista. Universidade de Brasília. Faculdade de Tecnologia. Brasília - DF, p. 161. 2008. Disponível em< http://bdtd.bce.unb.br/tedesimplificado/tde_busca/arquivo.php?codArquivo=3768>. Acesso em: 16 mai 2009.

DOLAN, K. Genetic Programming Source. Evolutionary Computing at Genetic Programming.us, 2009. Disponivel em: <http://www.geneticprogramming.us/Home_Page.html>. Acesso em: 19 jun. 2010.

EGGERMONT, J. Data Mining using Genetic Programming - Classification and Symbolic Regression. Institute for Programming research and Algorithmics, Leiden Institute of Advanced Computer Science, Faculty of Mathematics \& Natural Sciences, Leiden University. The Netherlands, p. 7. 2005. Disponível em< http://www.cs.bham.ac.uk/~wbl/biblio/gp-html/eggermont_thesis.html>. Acesso em: 23 abr. 2010.

FALCO, I. D.; CIOPPA, A. D.; TARANTINO, E. Discovering interesting classification rules with genetic programming. Applied Soft Computing. vol. 1, nr. 4, p. 257 - 269, 2002. ISSN "1568-4946", "DOI: 10.1016/S1568-4946(01)00024-2". Disponivel em: <http://www.sciencedirect.com/science/article/B6W86-44KWJTS-1/2/f24d9f9a27726e8b5982f5e4f0173a9d>. Acesso em: 28 dez. 2009.

FAN, Z. P. et al. A method for member selection of R&D teams using the individual and collaborative information. Expert Systems with Applications, vol. 36, nr. 4, p. 8313-8323, 2009. ISSN "0957-4174', "DOI:10.1016/j.eswa.2008.10.020". Disponivel em: <http://www.sciencedirect.com/science/article/B6V03-4TTMN95-2/2/11a4cbe20b97e457f5bbcb474dcfc98f>. Acesso em: 07 ago. 2009.

FAPEG. Resolução Normativa CONSUP n° 06/2007. Normas para credenciamento de Redes Goianas de Pesquisa. Conselho Superior da Fundação de Amparo à Pesquisa do Estado de Goiás, Goiânia, 2007. Disponivel em: <http://www.fapeg.go.gov.br/?p=286>. Acesso em: 5 abr. 2009.

FAPEG. Diretório de Redes Goianas de Pesquisa. Fundação de Amparo a Pesqusia do Estado de Goiás, 2010. Disponivel em: <http://www.fapeg.go.gov.br/gestor/indicadores_redes.php>. Acesso em: 10 Outubro 2010.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From Data Mining To Knowledge Discovery in Databases. AI Magazine. vol. 17, p. 37-54, 1996. Disponivel em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.1071>. Acesso em: 27 jul. 2010.

FIRAT, A.; CHATTERJEE, S.; MUSTAFA, Y. Genetic clustering of social networks using random walks. Computational Statistics & Data Analysis. vol. 51, nr. 12, p. 6285-6294, 2007. ISSN "0167-9473", "DOI: 10.1016/j.csda.2007.01.010". Disponivel em: <http://www.sciencedirect.com/science/article/B6V8V-4MYD60N-3/2/0a221d7407139e6c1e36706366e9cf21>. Acesso em: 07 ago. 2009.

FURLAN, J. D. Modelagem de Objetos Através da UML. 1ª. ed. [S.l.]: Makron Books, 1998. 352 p. ISBN 85-346-0924-1.

GOIÁS. Lei nº 15.472, de 12 de dezembro de 2005. Cria a Fundação de Amparo à Pesquisa do Estado de Goiás - FAPEG e dá outras providências. D.O. de 31-01-2006. Gabinete Civil da Governadoria,Superintendência de Legislação., 2005. Disponivel em: <http://www.gabinetecivil.goias.gov.br/leis_ordinarias/2005/lei_15472.htm>. Acesso em: 27 jul. 2010.

KOZA, J. R. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. Stanford University, Computer Science Department. Stanford, CA 94305: Margaret Jacks Hall, 1990. Disponivel em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.9961>. Acesso em: 27 abr. 2010.

KOZA, J. R. Concept Formation And Decision Tree Induction Using The Genetic Programming Paradigm. Stanford University, Computer Science Department, Stanford, California 94305 USA, 1991. Disponivel em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.5800>. Acesso em: 27 abr. 2010.

KOZA, J. R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. Sixth printing, 1998. ed. Cambridge, Massachusetts, US: Massachusetts Institute of Technology, 1992. 609 p. ISBN ISBN 0-262-11170-5.

KOZA, J. R. Genetic Algorithms and Genetic Programming, 2003. Disponivel em: <www.genetic-programming.com/c2003lecture1modified.ppt>. Acesso em: 20 fev. 2010.

LACERDA, E. G. M.; CARVALHO, A. C. Sistemas Inteligentes: Aplicações a Recursos Hídricos e Ciências Ambientais. In: _____ Introdução aos Algoritmos Genéticos. Porto Alegre, RS: Editora da Universidade Federal do Rio Grande do Sul, 1999. Cap. 3, p. 246.

MITRA, S.; ACHARYA, T. Data mining: multimedia, soft computing, and bioinformatics, p. 401, 2003. ISSN 0471460540. Disponivel em: <http://books.google.com/books?id=VPeOaKNfDlgC&pg=PA4&dq=kdd&hl=pt-BR#v=onepage&q=kdd&f=false>. Acesso em: 02 agosto 2009.

MUNI, D. P.; PAL, N. R.; DAS, J. A Novel Approach to Design Classifiers. IEEE Transactons on Evolution Computation, v. 8, n. 2, p. 183-195, Abril 2004. ISSN "1089-778X ", "DOI=10.1109/TEVC.2004.825567".

PAPPA, G. L. Seleção de Atributos Utilizando Algoritmos Genéticos Multiobjetivos. Tese de Mestrado, Pontifícia Universidade Católica do Paraná, 85 p., Curitiba, 2002. Disponivel em: <www.lania.mx/~ccoello/EMOO/thesis_pappa.pdf.gz>. Acesso em: 09 ago. 2009.

PENTAHO. Introducing the Pentaho BI Suite Community Edition. Pentaho Open Source Business Inteligence, 2009. Disponivel em: <http://wiki.pentaho.com/download/attachments/12386846/community_user_guide.pdf?version=1>. Acesso em: 11 nov. 2009.

QUINLAN, J. R. Induction of Decision Trees. Machine Learning 1, Kluwer Academic Publishers, Boston - Manufactured in the Netherlands, p.81-106, 1986. ISSN "0885-6125", "DOI=10.1023/A:1022643204877". Disponivel em: <http://www.springerlink.com/content/q30185t13m5n7000/>. Acesso em: 07 ago. 2010.

RIBAS, C. S. D. C.; ZIVIANI, P. Redes de informação: novas relações sociais. Revista de Economía Política de las Tecnologías de la Información y Comunicación. vol. X, n. 1, p. EPTIC On Line, 2008. Disponivel em: <http://www.eptic.com.br/arquivos/Revistas/v.%20X,n.%201,2008/ACludiaRibas-PaulaZiviani.pdf>. Acesso em: 27 jul. 2010. www.eptic.com.br.

ROMÃO, W.; FREITAS, A. A.; PACHECO, R. D. S. Uma revisão de abordagens genético-difusas para descoberta de conhecimento em banco de dados. Acta Scientiarum. v.22, n.5, p.1347-1359, 2000. ISSN "1415-6814". Disponivel em: <http://periodicos.uem.br/ojs/index.php/ActaSciTechnol/article/viewFile/3074/2219>. Acesso em: 18 out. 2009.

ROMERO, R.; MANTOVANI, J. R. S. Introdução a Metaheurísticas. Anais do 3o Congresso Temático de Dinámica e Controle da SBMAC, DEE-FEIS-UNESP-Ilha Solteira-Brasil, 2004.

SALMAN, A. K. D. Conceitos básicos de genética de populações. Embrapa Rondônia. Porto Velho, RO. p. 27, 2007. ISSN 0103-9865. Disponivel em: <http://www.cpafro.embrapa.br/portal/publicacao/117/>. Acesso em: 16 fev. 2010.

SIFAPS. 2a Reunião SIFAPs - São Paulo/SP - 18 e 19 de Junho. Sistema para Indicadores das FAPs, 2009. Disponivel em: <http://sifaps.egc.ufsc.br/index.php?option=com_content&view=article&id=71:reuniao-em-sao-paulo-18-e-19-de-junho&catid=31:reunioes&Itemid=40>. Acesso em: 10 Março 2010.

WITTEN, I. H.; FRANK, E. Data mining: practical machine learning tools and techniques, p. 525, 2005. ISSN 0120884070. Disponivel em: <http://books.google.com/books?id=QTnOcZJzlUoC&printsec=frontcover&dq=data+mining&hl=pt-BR#v=onepage&q=&f=false>. Acesso em: 02 agosto 2009.

ANEXO A

RESULTADO DO EXPERIMENTO II

Processando Descoberta da Regra de Classificação...

Inicializando conjunto de dados de treinamento...
Gerando população inicial...
Gerando regra de classificação...
Resultado...

Execução Nr.:	5
Conjunto de classes ou terminais:	79 101 392
Conjunto de atributos ou funções:	area equipe instparc grupocnpq
Número de argumentos dos atributos:	3 5 7 12
Conjunto de domínios dos atributos:	area :1 5 6 equipe :131 149 445 719 10007 instparc :2 5 128 250 296 315 378 grupocnpq :5 55 116 147 157 177 213 284 304 427 443 460
Tamanho da população inicia:	150
Quantidade de gerações:	50
Profundidade da árvore:	4

Melhor solução encontrada:

Geração:	35
Individuo nr:	23
Expressao:	(area (instparc (instparc 392 392 392 79 79 392 101) (instparc 79 392 392 79 101 101 392) (equipe 101 392 101 101 79) (instparc 392 101 79 79 392 101 101) (grupocnpq 79 79 101 101 79 392 79 392 101 392 79 101) (equipe 101 392 101 79 79) (instparc 79 392 392 79 101 101 392) (equipe (grupocnpq 79 101 101 79 101 392 101 79 392 392 101 79) (equipe 392 101 79 101 392) (equipe 392 101 101 101 79) (instparc 392 392 392 79 79 392 101) (grupocnpq 79 101 101 79 101 392 101 79 392 392 101 79) (instparc (instparc 79 392 392 79 101 101 392) (instparc 101 79 79 392 101 101 79) (instparc 101 79 79 392 101 101 79) (equipe 392 101 79 101 392) (instparc 392 392 392 79 79 392 101) (grupocnpq 79 392 79 79 101 79 392 79 79 79 101 101) (instparc 79 392 392 79 101 101 392))
Nr de casos de treinamento:	80
Nr de casos corretos (rawFitness):	80
Percentual de casos corretos:	100.00 %

Processando Classificação...
Inicializando conjunto de dados de teste...
Gravando resultado no banco de dados...

Resultado da Classificação:

Redes similares com pelo menos três atritubos comuns						
area	equipe	instituicao parceira	grupo cnpq	rede similar	rede classe	atributos comuns

6	96	2	5	41	79	3
1	114	2	304	57	392	3
1	763	2	304	57	392	3
1	236	2	304	156	392	3
1	236	250	304	156	79	3
5	445	5	5	174	101	4
5	445	5	443	174	101	4
5	445	1	5	174	101	3
5	445	1	443	174	101	3
5	445	210	5	174	101	3
5	445	210	443	174	101	3
5	445	263	5	174	101	3
5	445	263	443	174	101	3
5	445	2	460	174	101	4
5	445	2	304	174	101	4
5	445	5	460	174	101	4
5	445	5	304	174	101	4
5	445	1	460	174	101	3
5	445	1	304	174	101	3
5	445	210	460	174	101	3
5	445	210	304	174	101	3
5	445	263	460	174	101	3
5	445	263	304	174	101	3
5	445	2	5	174	101	4
5	445	2	443	174	101	4
6	236	2	304	267	79	3
6	285	2	5	271	79	3
6	336	2	177	328	79	3
6	336	5	177	328	79	3
1	343	2	304	332	392	3
1	706	2	157	377	392	3
1	706	5	157	377	392	3
6	870	2	304	399	79	3
1	878	2	177	407	392	3

Descrição dos atributos comuns relacionados às Redes: 79 101 392

idArea	Area da Agenda Goiana
1	Qualidade de Vida
5	Agronegócios, Desenvolvimento Rural e Fundiário
6	Pesquisa Inicial e Fundamental

idInst	Instituições Parceiras
2	Universidade Federal de Goiás
250	Secretaria de Estado da Saúde
5	Universidade Estadual de Goiás
1	Pontifícia Universidade Católica de Goiás
210	Ledal Quimica do Brasil
263	Secretaria Municipal de Saúde de Goiania

idGrupo	Grupos de Pesquisa do CNPq
5	Análise Farmacêutica e Ambiental - UFG
304	Núcleo de Estudos e Pesquisas Tóxico-Farmacológicas - NEPET-UFG
440	Grupo de Pesquisa em Plantas Medicinais e Fitoterápicos - UFG

460	Pesquisa, desenvolvimento e inovação tecnológica de produtos farmacêuticos, cosméticos e nutracêuticos a partir de matérias primas de origem natural, sintética ou biotecnológica. - UFG
177	Grupo de Pesquisa em Bioatividade Molecular - UFG
157	Grupo de Estudo e Pesquisa em Endocrinologia, Metabolismo e Nutrição e suas Implicações na Saúde - UFG

Descrição das Redes de Pesquisa com indicativo de similaridade:

idRede	Titulo Rede
41	Rede Goiana De Pesquisa Em Biossensores Para Monitoramento Ambiental
57	Rede Goiana De Pesquisa Monitorização Terapêutica, Toxicológica E Controle De Qualidade De Medicamentos
79	Rede Goiana de Pesquisa para Avaliação de Extratos de Espécies Vegetais do Bioma Cerrado quanto ao seu Potencial Citotóxico, Genotóxico e Farmacológico
101	Rede Goiana de Pesquisa em Cultivo e Controle de Qualidade de Plantas Medicinais
156	Rede Goiana De Pesquisa Vigilância À Saúde Do Idoso No Estado De Goiás
174	Rede Goiana De Pesquisa, Desenvolvimento E Inovação De Produtos Cosméticos E Farmacêuticos A Partir De Plantas Medicinais
267	Rede Goiana De Pesquisa Em Imunopatologia Da Doença De Alzheimer
271	Rede Goiana De Pesquisa Em Biodiversidade Microbiana Em Cerrado Goiano
328	Rede Goiana De Pesquisa Em Ressonância Magnética Nuclear
332	Rede Goiana De Pesquisa Em Toxicologia Pre-Clinica E Clinica
377	Rede Goiana De Pesquisa Em Doenças Cardiovasculares E Renais
392	Goiana de Pesquisa para Avaliação de Fatores Extrínsecos e Intrínsecos que Influenciam na Resposta Biológica a Fármacos e Correlatos
399	Rede Goiania De Pesquisa Em Biofarmácia E Farmacocinética
407	Rede Goiana De Pesquisa Em Epidemiologia E Caracterização De Agentes Etiológicos Causadores De Infecções Fúngicas Nosocomiais

ANEXO B

CÓDIGO ALGORITMO PG

```
% Implementa um Algoritmo PG para descoberta de regras de classificação,
% baseado no método de John R. Koza "Evolution of Classification:
% Decision-Tree Induction" (1990). Chama a função "GP_Class_KR_BD" para
% classificaçao do conjunto de dados de testes.
%
function GP_Tree_Class_KRn_BD(M,G,pMax,classe1,classe2,classe3,numrResult)

%----------------------Funções de Apoio ----------------------------------

   function [v] = eliminaDupl(z)
      %Dado um vetor elementos (z), a função "eliminaDupl" retorna
      %outro vetor (v) sem os elementos duplicados
      ok=0;
      k1=1;
      v(k1)=z(1);
      for k=2:length(z)
         e=z(k);
         for j=1:k1
            if e ==v(j)
               ok=1;
               break;
            end
         end
         if ok==0
            k1=k1+1;
            v(k1)=z(k);
         end
         ok=0;
      end
      clear e j ok z k k1;
   end

   function [tMax]=tamanhoMaximo
      %Calcula o tamanho maximo da árvore de decisão, ou expressão simbólica
      tMax = 0;
      for i=1:pMax
         tMax = tMax+(max(A))^(i-1);
      end
      clear i;
   end

   function [v_nivel]=qtdeNivel
      %Calcula a quantidade de níveis da árvore que podem ser usados para
      %aplicação dos operadores de cruzamento e mutaçao.
      z1=0; v_nivel=[];
      for z=2:pMax-1
         z1=z1+1;
         v_nivel(z1)=z;
      end
      clear z z1;
```

```
    end

%------------------Funções de inicialização------------------------------

    function [T]=iniT(x)
        %Inicializa conjunto de Terminais
        T=eliminaDupl(x);
    end


    function [D,A]=iniD(x)
        % Inicializa o conjunto de valore de domínios dos atributos (D)
        % Inicializa o conjunto de aridade dos atributos (A), que
        % corresponde ao número de argumentos de cada atributo.
        [n,m]=size(x);
        for b=1:m
            d=sort(eliminaDupl(x(:,b)));
            A(b)=length(d);
            D(1:A(b),b)=d(:);
        end
        clear b d n m x;
    end

    function leBD
        %Inicializa o conjunto de dados de treinamento a partir da
        %base de dado de Redes de Pesquisa.
        %
        %-Lê Banco de Dados
        fprintf('Inicializando conjunto de dados de treinamento... \n');

        % Set preferences with setdbprefs.
        s.DataReturnFormat = 'numeric';
        s.ErrorHandling = 'store';
        s.NullNumberRead = 'NaN';
        s.NullNumberWrite = 'NaN';
        s.NullStringRead = 'null';
        s.NullStringWrite = 'null';
        s.JDBCDataSourceFile = '';
        s.UseRegistryForSources = 'yes';
        s.TempDirForRegistryOutput = '';
        setdbprefs(s)

        % Estabelece a conexão com o banco de dados usando drive ODBC.
        conn = database('Mestrado','root','admin');

        % Lê dados do banco de dados por meio do comando "select".
        parm=['(',num2str(classe1),',',num2str(classe2),',',num2str(classe3),')'];
        parm1=('ORDER BY r.`idRedePesquisa` ASC');
            result = exec(conn,['SELECT DISTINCT t.`idAreaAgendaGoiana` AS
            idArea,p.`idPessoa`,i.`idInstituicao` AS idInstParc,pq.`idGrupoCNPq` AS
            idGrCNPq,r.`idRedePesquisa` AS idRede FROM `redespesquisa` r INNER JOIN `papeisexercidos` pe
            ON r.`idRedePesquisa` = pe.`idRedePesquisa`INNER JOIN `temasagendagoiana` t ON
            r.`idTemaAgendaGoiana` = t.`idTemaAgendaGoiana`INNER JOIN `papeisexercidos` pe1 ON
            r.`idRedePesquisa` = pe1.`idRedePesquisa`INNER JOIN `instituicoes` i ON pe1.`idInstituicao` =
            i.`idInstituicao`INNER JOIN `pessoas` p ON pe.`idPessoa` = p.`idPessoa`INNER JOIN
            `pessoas_grupos_cnpq` pq ON pe.`idPessoa` = pq.`idPessoa` WHERE pe.`idPapel` IN (1,2) and
            pe1.`idPapel` = 3 and r.`idRedePesquisa` in ',parm,parm1]);
        result = fetch(result);
        close(result)
```

```
    % Fecha conexão com o banco de dados.
    close(conn);

    % Atualiza conjunto de treinamento X
    X = result.Data;
  end

  function IniConjuntos
    %-Inicializa matriz do conjunto de treinamento(X)
%     [X,x] = xlsread('c:\\PGDadosTreinamento.xls',1);
    leBD;
    [nX,mX] = size(X);

    %-Inicializa vetor do conjunto de Funções (F)
    F = {'area' 'equipe' 'instparc' 'grupocnpq'};

    %-Inicializa vetor do conjunto de Terminais (T)
    T = sort(iniT(X(1:nX,mX)));

    %-Inicializa matriz do conjunto de Domínios e vetor de Aridades
    [D,A]=iniD(X(1:nX,1:mX-1));

    %-Calcula tamanho máximo da árvore
    [tMax]=tamanhoMaximo;

    %-Guarda a qtde de níveis da árvore possíveis de aplicação dos operadores
    %-de cruzamento e mutação
    [v_nivel]=qtdeNivel;

    %-Inicializa constantes: Populaçao inicial e Fitness Case
    pop_ini=M;
    FC=nX;
  end

  function [ind,indTree]=guardaNo(ind,indTree,no,ft,nivel)
    %Insere nó na árvore de indivíduos
    ind=ind+1;
    indTree(ind,1)= no;     %Indice de F
    indTree(ind,2)= ft;     %Indicador de função/terminal
    indTree(ind,3)= nivel;  %Nível da árvore
  end

  function [ind,indTree]=geraFull(indTree,prof,nivel,profMax,ind)
    %Gera a árvore de indivíduos, randomicamente, usando o
    %conjunto de funções F e o conjunto de terminais T, pelo método
    %Full, descrito por Kevin Dollan em
    %http://www.geneticprogramming.us/The_Initial_Population.html

    %Se não é a profundidade máxima, escolha uma função
    if(prof>1)
      no = ceil(length(F)*rand(1,1));
      %Insere nó função
      ft =1;
      [ind,indTree]=guardaNo(ind,indTree,no,ft,nivel);
      %Recursivamente preencha os nós filhos
      for i1 = 1: A(no);
        [ind,indTree] = geraFull(indTree,prof-1,nivel+1,profMax,ind);
      end
    % Senão, escolhe um terminal
    else
```

```
        %Insere nó terminal
        no=ceil((length(T))*rand(1,1));
        ft =2;
        [ind,indTree]=guardaNo(ind,indTree,no,ft,nivel);
      end
    end

    function [Y]=geraPopIni
      %Gera população inicial de indivíduos (Y)
      disp ('Gerando população inicial...');
      ind=0; indTree=[]; nivel=1; prof = pMax; profMax=pMax;
      for i=1:M
        [ind,indTree]=geraFull(indTree,prof,nivel,profMax,ind);
        for i2=1:length(indTree(:,1))
          Y(i,i2,:)= indTree(i2,:);
        end
        indTree=[]; ind = 0;
      end
      clear i i2 ind prof;
    end

    function [ind_dupl]=avalia_indiv_dupl(Y,M,m)
      %-Avalia população a fim de encontrar indivíduos duplicados
      k=0; ind_dupl=[];
      for i=1:M
        y1(:,1)=Y(i,:,1);
        for j=i+1:M
          y2(:,1)=Y(j,:,1);
          if (length((find(y1(:)==y2(:)==1)))==m)
            k=k+1;
            ind_dupl(k)=j;
          end
        end
      end
      clear k i y1 y2 j k;
    end

%------------------Funções de resolução da árvore-------------------------

    function [i]=avaliaNo(y,x,i,i1)
      %Avalia o primeiro argumento de um nó tipo função.
      % Se vlrY=vlrX
      %    retorna o índice do primeiro argumento do atributo da função
      % Senão
      %    chama a funçao matlab "buscaRamo" para buscar o argumento que
      %    corresponde ao elemento do conjunto de treinamento, a fim de
      %    identificar o ramo a ser seguido. Retorna o índice deste ramo
      i_no=y(i,1);
      nrArg=1;
      vlrY=D(nrArg,i_no);
      vlrX=x(i_no);
      if vlrX==vlrY                    %vlrX=vlrY
        i=i+1;
      else
        i=buscaRamo(y,i,vlrX,i1,nrArg); %vlrX~=vlrY
      end
      clear i1 nrArg i_no vlrX vlrY;
    end

    function [i]=buscaRamo(y,i,vlrX,i1,nrArg)
```

```
   %Avalia os próximos argumento de um nó tipo função,
   %a fim de identificar o ramo a ser seguido.
   %Retorna o índice deste ramo
   z=0;
   while (z==0)
      nrArg=nrArg+1;
      i_no=y(i,1);
      vlrY=D(nrArg,i_no);
      if vlrX==vlrY
         i= nrRamo(y,i,i1,nrArg);
         z=1;
      end
   end
   clear z i1 nrArg i_no vlrX vlrY;
end

function [i]=nrRamo(y,i,i1,nrArg)
   %Busca o índice do ramo árvore a ser seguido
   inicio = i;
   nivel_busca=i1+1;
   k=0;
   v=[];
   for j=inicio:length(find(y(:,1)>0))
      if (y(j,3)==nivel_busca)
         k=k+1;
         v(k)=j;
         if (k==nrArg)
            i=v(k);
            return;
         end
      end
   end
   clear inicio nivel_busca k v j;
end

function [S]=solvTree(y,x)
   %Faz um caminhamento pré-fixado na árvore, a fim de percorrer o
   %ramo da árvore, da raíz às folhas, para avaliar a regra.
   %Cada elemento de x é avaliado, em cada nível da árvore y, se o nó
   %é uma função, o valor do atributo é novamente avaliado. Se o nó é
   %um terminal, o valor do terminal é retornado em uma variável (S).
   i=1; S=0;
   for i1=1:pMax
      if (y(i,2)==1)      %Se função
         [i]=avaliaNo(y,x,i,i1);
      else              %Se terminal
         S=T(y(i,1));
      end
   end
   clear i i1;
end

function [aF,rF,AF,RF,a,y]=avaliaPop(Y)
   %-Avaliar cada indivíduo da população e atribuir-lhe um valor de fitness,
   %que expressa o quanto a regra está próxima da solução ideal. Este passo
   %envolve aplicar cada regra (y1...ym) sobre cada um dos elementos do
   %conjunto de treinamento (x1...xn), a fim de calcular o valor de fitness.
   %
   %FC = Nr de casos de treinamento
   %RF = Nr de casos de treinamento classificados dentro das classes corretas.
```

```
    %SF = FC-RF
    %AF = 1/(1+SF)

    [M,m,p]=size(Y);
    aF=[]; rF=[];

    %-Avalia indivíduos duplicados
    ind_dupl=avalia_indiv_dupl(Y,M,m);

    %-Para cada indivíduo da população, faça:
    for a = 1:M
      %Se indivíduo duplicado, ignorar indivíduo
      if find(ind_dupl==a)>0
      %Senão, avaliar indivíduo
      else
        y(1:m,1:p)=Y(a,1:m,1:p);
        RF = 0;
        for a2 = 1:nX
          x = X(a2,1:mX-1);
          S=solvTree(y,x);    %S=Classe do indivíduo
          class=X(a2,mX);     %class=Classe do dado de treinamento
          if S==class
            RF = RF + 1;    %Adiciona um ao raw fitness (RF)
          end
        end
        SF=FC-RF;             %Calcula fitness padronizada (SF)
        AF=1/(1+SF);          %Calcula fitness ajustada (AF)
        %Avalia fitness
        if (AF==1)            %Encontrado o melhor indivíduo ou melhor regra
          break;
        else
          rF(a)=RF;
          aF(a)=AF;
        end
      end
    end
    clear ind_dupl a2 S class k SF;
  end

%---------------Funções de fitness e seleção ------------------------------

  function [NF]=normalizedFitness(aF)
    %Calcula a aptidão normalizada de todos os indivíduos
    for k = 1:length(aF)
      NF(k)= aF(k)/sum(aF);
    end
    clear k;
  end

  function [v]=melhorFitness(z)
    %Busca os melhores valores de normalized fitness (NF)
    ok=0;
    k1=1;
    v(k1)=z(end);
    for k=length(z):-1:1
      e=z(k);
      for j=1:k1
        if e==v(j)
          ok=1;
          break;
```

```
            end
        end
        if ok==0
            k1=k1+1;
            v(k1)=z(k);
        end
        ok=0;
    end
    clear ok k1 z k e;
end

function [YM,melhorNF]=selecaoElitista(Y,NF)
    %Seleciona os melhores indivíduos pelo método elitista, sendo
    %selecionados os indivíduos correspondentes aos 3 melhores
    %valores de fitness. Estes indivíduos são copiados para a próxima
    %geração pelo princípio da seleção natural e sobrevivência do
    %melhor.

    clear YM I i_NF;
    YM = [];
    [NF,I]=sort(NF);              %classifica o vetor NF
    MF = melhorFitness(NF);       %MF=Melhores indivíduos
    melhorNF=I(end);              %Índice do melhor indivíduo
    k1=0; k2=1;
    c=MF(k2);
    k3=length(NF);
    while k3>=1
        if (NF(k3)==c)
            k1=k1+1;
            YM(k1,:,:)=Y(I(k3),:,:);
            k3=k3-1;
        else
            k2=k2+1;
            if k2>3
                clear MF k1 k2 c k3;
                return;
            end
            c=MF(k2);
        end
    end
    clear MF k1 k2 c k3;
end

%----------------------Funções de cruzamento-------------------------------

function [ini,fim,tam_no,ultimo_no,qtde_no]=get_no_cros(y,nivel)
    %Seleciona nó da árvore, aleatoriamente. Retorna informações sobre
    %o nó selecionado: posição inicial, posição final,tamanho do bloco,
    %indicador de último nó da árvore e a qtde de nós da árvore.
    %
    tam=length(find(y(:,1)>0));     %tamanho real do individuo
    v=find(y(:,3)==nivel);          %vetor de nós do nível da árvore
    qtde_no =length(v);             %quantidade de nós do nível selecionado
    ultimo_no=0;
    i_v_ini=ceil(length(v)*rand(1,1));
    ini=v(i_v_ini);                 %Posiçao ínicial indivíduo
    if (i_v_ini==length(v))
        fim=tam;                    %Posiçao final indivíduo
        ultimo_no=1;                %Indicador de último nó
    else
```

```
      fim=v(i_v_ini+1)-1;
      if y(fim,3)< nivel
        fim=fim-1;
      end
    end
    tam_no=fim-ini+1;
    clear v i_v_ini tam;
end

function [y_temp]=cruza_individuo1(y1,y2,ini1,ini2,fim1,tam2,ultimo_no)
    %Faz a troca de material genético no indivíduo 1
    if (length(find(y1(:,1)>0))-(fim1-ini1)+tam2) > tMax
      tMax = (length(find(y1(:,1)>0))-(fim1-ini1)+tam2);
    end
    clear y_temp;
    y_temp = zeros(tMax,3);
    y_temp(1:ini1-1,:)=y1(1:ini1-1,:);  %Copia 1ª parte do indivíduo
    j1=ini1;
    j2=ini2;
    j=1;
    while (j<=tam2)
      y_temp(j1,:)=y2(j2,:);
      j1=j1+1;
      j2=j2+1;
      j=j+1;
    end
    if (ultimo_no == 0)              %Copia o restante dos elementos
      j=length(find(y_temp(:,1)>0))+1;
      j1=fim1+1;
      while ((j<=tMax) && (j1<=tMax))
        y_temp(j,:)=y1(j1,:);
        if y_temp(j,3)==0
          break;
        end
        j=j+1;
        j1=j1+1;
      end
    end
    clear y1 y2 j1 j2 j;
end

function [y_temp]=cruza_individuo2(y1,y2,ini1,ini2,fim2,tam1,ultimo_no)
    %Faz a troca de material genético no indivíduo 2
    if (length(find(y2(:,1)>0))-(fim2-ini2)+tam1) > tMax
      tMax = (length(find(y2(:,1)>0))-(fim2-ini2)+tam1);
    end
    clear y_temp;
    y_temp = zeros(tMax,3);
    y_temp(1:ini2-1,:)=y2(1:ini2-1,:);   %Copia 1ª parte do indivíduo
    j1=ini1;
    j2=ini2;
    j=1;
    while (j<=tam1)
      y_temp(j2,:)=y1(j1,:);
      j1=j1+1;
      j2=j2+1;
      j=j+1;
    end
    if (ultimo_no == 0)              %Copia o restante dos elementos
      j=length(find(y_temp(:,1)>0))+1;
```

```
        j1=fim2+1;
        while ((j<=tMax) && (j1<=tMax))
            y_temp(j,:)=y2(j1,:);
            if y_temp(j,3)==0
                break;
            end
            j=j+1;
            j1=j1+1;
        end
    end
    clear y1 y2 j1 j2 j;
end

function [Y1,i]=nova_pop(y1_temp,y2_temp,Y1,i)
    %Copia indivíduos para a nova geração temporária
    if isempty(Y1)==1
      i=1;
    else
      i=length(Y1(:,1,1))+1;
    end
    for i2=1:length(y1_temp)
        Y1(i,i2,:)= y1_temp(i2,:);
    end
    i=i+1;
    for i2=1:length(y2_temp)
        Y1(i,i2,:)= y2_temp(i2,:);
    end
   clear i2;
end

function [Y1]=crossover(YM)
    %Do conjunto de melhores indivíduos são selecionados dois
    %indivíduos pais, para serem submetidos ao operador de cruzamento.
    %Para cada indivíduo do conjunto de melhores indivíduos é selecionado
    %outro par aleatoriamente. Um ponto aleatório de cruzamento é
    %escolhido em cada indivíduo-pai e as árvores abaixo destes pontos
    %são trocadas.

    %Inicializa variáveis
    clear Y1; Y1=[];
    [n,m,p]=size(YM);
    y1 = zeros(tMax,1);
    y2 = zeros(tMax,1);

    i=0;
    for c1=1:n
        %seleciona nível aleatoriamente
        if length(v_nivel) == 1
            nivel=v_nivel(1);
        else
            nivel=v_nivel(ceil(length(v_nivel)*rand(1,1)));
        end

        %Seleciona indivíduo 1 e posição do nó para cruzamento
        y1(1:m,1:p)=YM(c1,1:m,1:p);
        [ini1,fim1,tam1,ultimo_no1,qtde_no]=get_no_cros(y1,nivel);

        %Seleciona indivíduo 2 e posição do nó para cruzamento
        c2 = ceil(n*rand(1,1));
        y2(1:m,1:p)=YM(c2,1:m,1:p);
```

```
        [ini2,fim2,tam2,ultimo_no2,qtde_no]=get_no_cros(y2,nivel);

        %Valida segundo indivíduo.
        %Se indivíduos iguais e único nó, seleciona outro indivíduo
        while ((c1==c2) && (qtde_no==1)) || ((length(find(y1(:,1)==y2(:,1)))==m) && (qtde_no==1))
          c2 = ceil(n*rand(1,1));
          y2 = zeros(tMax,1);
          y2(1:m,1:p)=YM(c2,1:m,1:p);
          [ini2,fim2,tam2,ultimo_no2,qtde_no]=get_no_cros(y2,nivel);
        end
        %Se indivíduos iguais e mesma posição do cruzamento, gera uma nova
        %posição para o segundo indivíduo.
        while (c1==c2)&&(ini1==ini2)
          [ini2,fim2,tam2,ultimo_no2,qtde_no]=get_no_cros(y2,nivel);
        end

        %Cruza indivíduos
        [y1_temp]=cruza_individuo1(y1,y2,ini1,ini2,fim1,tam2,ultimo_no1);
        [y2_temp]=cruza_individuo2(y1,y2,ini1,ini2,fim2,tam1,ultimo_no2);

        %Copia indivíduos para nova população
        [Y1,i]=nova_pop(y1_temp,y2_temp,Y1,i);
        if i>pop_ini-n
          return
        end
      end
      clear n m p y1 y2 i c1 c2 y1_temp y2_temp ini1 ini2
      clear fim1 fim2 tam1 tam2 ultimo_no1 ultimo_no2 qtde_no
    end

%----------------------Funções de mutação-------------------------------

    function [pos_mut]=analisaPathNoMut(y1)
      %Analisa se existem nós iguais ao nó raiz no caminho da árvore
      tam=length(find(y1(:,1)>0));
      pos_mut=[];
      raiz=y1(1,1);
      z1=0;
      for z=2:tam
        if y1(z,2)==1 && y1(z,1)== raiz
          z1=z1+1;
          pos_mut(z1,1)=z;
          pos_mut(z1,2)=y1(z,3);
        end
      end
      clear tam raiz z z1;
    end

    function [y1,mut]=mutacaoPath(y1)
      %verifica se o nó raiz é repetido no caminho que vai da raiz à
      %folha da árvore, se houver, um novo ramo é gerado a partir do
      %ponto identificado.
      mut=0;
      [pos_mut]=analisaPathNoMut(y1);
      if isempty(pos_mut)==0
        c1=0;
        while c1==0
          %Gera novo ramo para a árvore
          nv=pos_mut(1,2);
          ind=0; y2=[]; prof = pMax-nv+1; profMax= prof; ini2=1;
```

```
        [tam2,y2]=geraFull(y2,prof,nv,profMax,ind);

        %Busca informações do indivíduo para mutação
        ini1=pos_mut(1,1);
        fim1=length(find(y1(:,1)>0));
        ultimo_no=1;
        for c2=ini1+1:length(find(y1(:,1)>0))
           if y1(c2,3)== nv
              fim1=c2-1;
              ultimo_no=0;
              break;
           end
        end

        %Efetua mutação no indivíduo selecionado
        [y1]=cruza_individuo1(y1,y2,ini1,ini2,fim1,tam2,ultimo_no);
        mut=1;
        [pos_mut]=analisaPathNoMut(y1);
        if isempty(pos_mut)==1
          break;
        end
      end
   end
   clear y2 nv ultimo_no c2 c3 pos_mut ind prof tam ini1 fim1 ini2 tam2
end

function [ind_dupl]=avalia_ramo_dupl(y_temp)
   %-Avalia o indivíduo, a fim de encontrar ramos duplicados
   ind_dupl=0;
   [qlin,qcol]=size(y_temp);
   for i=1:qlin
      y1(:,1)=y_temp(i,:);
      for j=i+1:qlin
        y2(:,1)=y_temp(j,:);
        if (length((find(y1(:)==y2(:)==1)))==qcol)
          ind_dupl=j;
          break;
        end
      end
   end
   clear qlin qcol i j y1 y2;
end

function [ind_dupl]=analisaNoMut(y1)
   %Busca posição do nó para mutação

   tam=length(find(y1(:,1)>0));
   y2=[];
   v=find(y1(:,3)==2);
   for z = 1:length(v)
      ini=v(z);
      if z==length(v)
        tam_no=tam-ini+1;
      else
        tam_no=v(z+1)-ini;
      end
      for z1=1:tam_no
         y2(z,z1)=y1(ini,1);
         ini=ini+1;
      end
```

```
   end
   ind_dupl=avalia_ramo_dupl(y2); %-Avalia ramos duplicados
   clear y2 tam v z tam_no ini z1;
end

function [y1,mut]=mutacaoDupl(y1,mut)
   %O indivíduo é analisado qto à existência de ramos iguais ou
   %duplicados no segundo nível da árvore, sendo gerado um novo ramo.

   [pos_mut]=analisaNoMut(y1);
   if pos_mut>0
     %Gera novo ramo para a árvore
      nv=2; ind=0; y2=[]; prof = pMax-nv+1; profMax= prof;
      [ind,y2]=geraFull(y2,prof,nv,profMax,ind);

      %Busca informações do indivíduo para mutação
       tam=length(find(y1(:,1)>0));
       v=find(y1(:,3)==nv);
       ini1=v(pos_mut);
       ultimo_no=0;
       if (pos_mut==length(v))
          fim1=tam;
          ultimo_no=1;
       else
          fim1=v(pos_mut+1)-1;
       end
       ini2=1; tam2=length(y2(:,1));

      %Efetua mutação no indivíduo selecionado
       [y1]=cruza_individuo1(y1,y2,ini1,ini2,fim1,tam2,ultimo_no);
       mut=1;
   end
   clear y2 nv ultimo_no pos_mut ind prof tam v ini1 fim1 ini2 tam2
end

function [Y1]=mutacao(Y1)
   %Aplica o operador de mutação, que foi adotado visando melhoria da
   %qualidade da regra.
   [n,m,p]=size(Y1);
   y1 = zeros(tMax,1);
   for c1=1:n
      y1(1:m,1:p)=Y1(c1,1:m,1:p);

      %verifica se o nó raiz é repetido no caminho que vai da raiz à
      %folha da árvore, se houver, um novo ramo é gerado a partir do
      %ponto identificado.
      [y1,mut]=mutacaoPath(y1);

      %verifica se há ocorrência de ramos duplicados, sendo gerados
      %novos ramos para os pontos identificados
      [y1,mut]=mutacaoDupl(y1,mut);

      %Se ocorreu mutaçao, guarda indivíduo na nova população
      if mut==1
        for c4=1:length(y1)
            Y1(c1,c4,:)= y1(c4,:);
        end
      end
   end
   clear n m p c1 mut c4;
```

```
  end

%---------------------Funções de finalizaçao-----------------------------

  function [Y1]=insereNovaGeracao(YM,Y1)
    %A nova população é composta dos indivíduos pais, selecionados pelo
    %método elitista mais os indivíduos filhos gerados pelas operações
    %de cruzamento e de mutação

    m=length(Y1(:,1,1));
    for k=1:length(YM(:,1,1))
       m=m+1;
       for k1=1:length(YM(k,:,:))
          Y1(m,k1,:)= YM(k,k1,:);
       end
    end
    clear m k k1;
  end

  function [tree]=montaExpr(y)
    %Monta a expressão simbólica para apresentação no relatório final

    T1 = num2cell(T);
    abreP={'('};
    fechaP={')'};
    b=1;
    b3=0;
    while b<=length(find(y(:,1)>0))
       b2=y(b,1);
       if y(b,2)==1
         b3=b3+1;
         tree(b3)= abreP;
         b3=b3+1;
         tree(b3)=F(b2);
       else
         b3=b3+1;
         tree(b3)=T1(b2);
         if y(b+1,2)==1
           b3=b3+1;
           tree(b3)= fechaP;
         end
       end
       b=b+1;
    end
    b3=b3+1;
    tree(b3)= fechaP;
    tree(b3+1)= fechaP;
    clear T1 abreP fechaP b b2 b3;
  end

  function [msg,rF_m,pRF,melhorNF,pop_m,stop]=condParadaMelhorSol(AF,a)
     %-Prepara apresentação da melhor solução encontrada

     stop=0; msg=' '; rF_m=0; pRF=0; melhorNF=0; pop_m=[];
     if (AF==1)
        msg='Melhor solução encontrada:';
        rF_m=RF;
        pRF=(rF_m/FC)*100;
        melhorNF=a;
        pop_m=montaExpr(y);
```

```
        stop=1;
      end
    end

    function [msg,rF_m,pRF,melhorNF,pop_m,stop]=condParadaSolAprox(g,n,m,p,melhorNF,Y)
      %-Prepara apresentação da solução aproximada encontrada

      stop=0; msg=' '; rF_m=0; pRF=0; pop_m=[];
      if (g==G)||(n >= pop_ini)
        msg=('Melhor solução aproximada encontrada:');
        rF_m=rF(melhorNF);
        pRF=(rF_m/FC)*100;
        y=[]; y(1:m,1:p)=Y(melhorNF,1:m,1:p);
        [pop_m]=montaExpr(y);
        stop=1;
      end
    end

    function mostraSolucao(msg,g,melhorNF,pop_m,FC,rF_m,pRF)
      %Apresenta o relatório final

      %Monstra os parâmetros do experimento

fprintf('_____
_______ \n');
      fprintf('\n');
      fprintf('Execução Nr.                    : %d \n',numrResult);
      fprintf('Conjunto de classes ou terminais  : ');
      for i=1:length(T)
        t=T(i);
        fprintf('%d ',t);
      end
      fprintf('\n');
      fprintf('Conjunto de atributos ou funções  : ');
      for i=1:length(F)
        f=char(F(i));
        fprintf('%s ',f);
      end
      fprintf('\n');
      fprintf('Número de argumentos dos atributos: ');
      for i=1:length(A)
        a=A(i);
        fprintf('%d ',a);
      end
      fprintf('\n');
      fprintf('Conjunto de domínios dos atributos: \n');
      [n,m]=size(D);
      for i=1:m
        f=char(F(i));
        fprintf('   %s :',f);
        for j=1:n
          d=D(j,i);
          if d>0
            fprintf('%d ',d);
          end
        end
        fprintf('\n');
      end
      fprintf('\n');
      fprintf('Tamanho da população inicial      : %d \n',pop_ini);
```

```
        fprintf('Quantidade de gerações          : %d \n',G);
        fprintf('Profundidade da árvore          : %d \n',pMax);

fprintf('_____________________________________________________________________________________________');

        %Monstra a solução encontrada
        fprintf('\n');
        fprintf('%s \n',msg);
        fprintf('Geração                        : %d \n',g);
        fprintf('Individuo nr                    : %d \n',melhorNF);
        fprintf('Expressao                       : ');
        for i=1:length(pop_m)
           if iscellstr(pop_m(i))==1
              melhor=char(pop_m(i));
              fprintf('%s ',melhor);
           else
              melhor=pop_m{i};
              fprintf('%d ',melhor(1));
           end
        end
        fprintf('\n');
        fprintf('Nr de casos de treinamento      : %d \n',FC);
        fprintf('Nr de casos corretos (rawFitness): %d \n',rF_m);
        fprintf('Percentual de casos corretos    : %3.2f %% \n',pRF);
        fprintf('_____________________________________________________________________ \n');
    end

    function mostraEvolucaoRegra(ER)
         %Mostra evolução da regra de classificação
         fprintf('\n');
         fprintf('Evolução da regra de classificação\n');
         fprintf('geração  melhor fitness  casos corretos \n');
         fprintf('---------------------------------------- \n');
         for i=1:length(ER(:,1));
            fprintf('   %3d         %3.4f          %3d',i,ER(i,1),ER(i,2) );
            fprintf('\n');
         end
         figure(1),hold on;
         for i=1:length(ER(:,1));
             plot(i,ER(i,1),'--r.','LineWidth',4);
         end
         figure(2),hold on;
         for i=1:length(ER(:,1));
             plot(i,ER(i,2),'--r.','LineWidth',4);
         end
    end


%---------------------------Corpo do Programa ----------------------------
%
%Inicializa variáveis globais
clc; format short
global X F T A D nivel tMax profMax pop_ini v_nivel nX mX FC;
fprintf('Processando Descoberta da Regra de Classificação... \n');
fprintf('------------------------------------------------- \n');

%1.Inicializar os parâmetros de execução do algoritmo: conjunto de
%treinamento (X), conjunto de funções (F), conjunto de terminais (T),
%conjunto de domínios (D), conjunto de aridade (A), tamanho e
%profundidade da Árvore
```

```
 IniConjuntos;

%2.Gerar uma população inicial de composição randômica de funções e
%terminais do problema, formando a descrição da regra ou indivíduo
 [Y]=geraPopIni; ER=[];
 fprintf('Gerando regra de classificação... \n');

%3.Executar iterativamente até que a quantidade de gerações seja atendida
for g=1:G
   %3.1 Avaliar cada indivíduo da população e atribuir-lhe um valor
   %de fitness, que expressa o quanto o quanto a regra está próximo da solução ideal
   [aF,rF,AF,RF,a,y]=avaliaPop(Y);

   %3.2 Verificar se a melhor regra foi encontrada. Se encontrar, sair do loop
   [msg,rF_m,pRF,melhorNF,pop_m,stop]=condParadaMelhorSol(AF,a);
   if (stop==1)
      ER(g,1)= aF(melhorNF); ER(g,2)= rF(melhorNF);
     break;
   end

   %3.3 Selecionar indivíduos para reprodução.
   %3.3.1 Calcular o valor de NF
   NF=normalizedFitness(aF);

   %3.3.2 Aplicar o método de seleção elitista
   [YM,melhorNF]=selecaoElitista(Y,NF);
    ER(g,1)= aF(melhorNF); ER(g,2)= rF(melhorNF);


   %3.4. Verificar se a melhor regra aproximada foi encontrada pela
   %convergência do algoritmo. Se encontrar, sair do loop
   [n,m,p]=size(YM);
   [msg,rF_m,pRF,melhorNF,pop_m,stop]=condParadaSolAprox(g,n,m,p,melhorNF,Y);
   if (stop==1)
      break;
   end

   %3.5. Aplicar o operador de cruzamento, pela recombinação genética de
   %dois indivíduos pais, que são selecionados do conjunto de melhores indivíduos
   [Y1]=crossover(YM);

   %3.6. Aplicar o operador de mutação para corrigir o problema de
   %repetição de características no indivíduo, melhorando a qualidade de sua formação
   [Y1]=mutacao(Y1);

   %3.7. Copiar os indivíduos gerados para a nova população
   [Y1]=insereNovaGeracao(YM,Y1);
   clear Y y aF rF pop;
   Y=Y1;
end

%4.Mostrar a solução encontrada
disp('Resultado...');
mostraSolucao(msg,g,melhorNF,pop_m,FC,rF_m,pRF);
mostraEvolucaoRegra(ER);

%Executa Algoritmo Classificador
GP_Class_KR_BD(y,T,D,pMax,classe1,classe2,classe3,numrResult);

end
```

ANEXO C

CÓDIGO ALGORITMO CLASSIFICADOR

```
% Implementa um algoritmo para classificar a base de dados, segundo a regra
% descoberta pela função "GP_Tree_Class_KRn_BD".
%
function GP_Class_KR_BD(y,T,D,pMax,rede1,rede2,rede3,numrResult)

   function [i]=avaliaNo(y,x,i,i1)
      %Avalia o primeiro argumento de um nó tipo função.
      % Se vlrY=vlrX
      %    retorna o índice do primeiro argumento do atributo da função
      % Senão
      %    chama a funçao matlab "buscaRamo" para buscar o argumento que
      %    corresponde ao elemento do conjunto de treinamento, a fim de
      %    identificar o ramo a ser seguido. Retorna o índice deste ramo
      %
      [i_no]=y(i,1);
      nrArg=1;
      vlrY=D(nrArg,i_no);
      vlrX=x(i_no);
      if vlrX==vlrY                    %vlrX=vlrY
         i=i+1;
      else
         i=buscaRamo(y,i,vlrX,i1,nrArg); %vlrX~=vlrY
      end
   end

   function [i]=buscaRamo(y,i,vlrX,i1,nrArg)
      %Avalia os próximos argumento de um nó tipo função, a fim de
      %identificar o ramo a ser seguido. Retorna o índice deste ramo
      %
      z=0;
      while (z==0)
         nrArg=nrArg+1;
         [i_no]=y(i,1);
         if (nrArg>length(D(:,i_no))) || (D(nrArg,i_no)==0)
            i=0;
            return;
         end
         vlrY=D(nrArg,i_no);
         if vlrX==vlrY
            i= nrRamo(y,i,i1,nrArg);
            z=1;
         end
      end
   end

   function [i]=nrRamo(y,i,i1,nrArg)
      %Busca o índice do ramo árvore a ser seguido
      inicio = i;
      nivel_busca=i1+1;
      k=0;
```

```
    v=[];
    for j=inicio:length(find(y(:,1)>0))
        if (y(j,3)==nivel_busca)
            k=k+1;
            v(k)=j;
            if (k==nrArg)
                i=v(k);
                return;
            end
        end
    end
end

function [S]=solvTree(y,x)
    %Faz um caminhamento pré-fixado na árvore, a fim de percorrer o
    %ramo da árvore, da raíz às folhas, para avaliar a regra.
    %Cada elemento de x é avaliado, em cada nível da árvore y, se o nó
    %é uma função, o valor do atributo é novamente avaliado. Se o nó é
    %um terminal, o valor do terminal é retornado em uma variável (S).

    i=1; S=0;
    for i1=1:pMax
        if (y(i,2)==1)      %Se função
            [i]=avaliaNo(y,x,i,i1);
        else
            S=T(y(i,1));
        end
        if i==0
          break;
        end
    end
end

function [i]=busca(k,v)
   %Busca o elemento "k" no vetor "v"
   i=0;
   for i1=1:length(v)
      if k==v(i1)
         i=1;
         break;
      end
   end
end

function [X]=leBD
    %-Lê Banco de Dados e inicializa o conjunto de dados de testes
    fprintf('Inicializando conjunto de dados de teste... \n');

    % Seta preferencias com setdbprefs.
    s.DataReturnFormat = 'numeric';
    s.ErrorHandling = 'store';
    s.NullNumberRead = 'NaN';
    s.NullNumberWrite = 'NaN';
    s.NullStringRead = 'null';
    s.NullStringWrite = 'null';
    s.JDBCDataSourceFile = '';
    s.UseRegistryForSources = 'yes';
    s.TempDirForRegistryOutput = '';
    setdbprefs(s)
```

```
   % Estabelece conexão com o banco de dados.
   % Using ODBC driver.
   conn = database('Mestrado','root','admin');

   % Le dados do banco de dados por meio do comando SQL "selec".
      result = exec(conn,'SELECT DISTINCT t.`idAreaAgendaGoiana` AS
      idArea,p.`idPessoa`,i.`idInstituicao` AS idInstParc,pq.`idGrupoCNPq` AS
      idGrCNPq,r.`idRedePesquisa` AS idRede FROM `redespesquisa` r INNER JOIN `papeisexercidos` pe
      ON r.`idRedePesquisa` = pe.`idRedePesquisa`INNER JOIN `temasagendagoiana` t ON
      r.`idTemaAgendaGoiana` = t.`idTemaAgendaGoiana`INNER JOIN `papeisexercidos` pe1 ON
      r.`idRedePesquisa` = pe1.`idRedePesquisa`INNER JOIN `instituicoes` i ON pe1.`idInstituicao` =
      i.`idInstituicao`INNER JOIN `pessoas` p ON pe.`idPessoa` = p.`idPessoa`INNER JOIN
      `pessoas_grupos_cnpq` pq ON pe.`idPessoa` = pq.`idPessoa` WHERE pe.`idPapel` IN (1,2) and
      pe1.`idPapel` = 3 ORDER BY r.`idRedePesquisa` ASC');
   result = fetch(result);
   close(result)

   % Fecha conexão com banco de dados.
   close(conn);

   % Inicializa o conjunto de dados de testes
   X = result.Data;
end

function [x2]=mapeamento(x1)
   %Mapear todos os valores dos atributos preditivos de (X) que
   %pertencem a (D)

   [nx1,mx1]=size(x1);
   x2=[];
   for b=1:nx1
      for b1=1:mx1-2
         k=x1(b,b1);
         tamD=length(find(D(:,b1)>0));
         v=D(1:tamD,b1);
         if busca(k,v)==1
            x2(b,b1)=1;
         else
            x2(b,b1)=0;
         end
      end
   end
   clear k tamD v nx1 mx1 b b1;
end

function [x3]=qtdeAtribMap(x1,x2,mX)
   %Verificar a quantidade de atributos preditivos mapeados e criar
   %uma nova coluna em (X) com a quantidade de atributos encontrados

   t=mX+2;
   t1=t+1;
   b1=0;
   x3=[];
   [nx1,mx1]=size(x1);
   for b=1:nx1
      x1(b,t)=length(find(x2(b,:)>0));
      x1(b,t1)=numrResult;

      if x1(b,t)>2
         b1=b1+1;
```

```
        x3(b1,:)=x1(b,:);
      end
   end
   clear t t1 b b1
end

function gravaResultBD(x1)
   %Grava resultado no Banco de Dados
   disp('Gravando resultado no banco de dados...');

   % Seta Preferencias com setdbprefs.
   s.DataReturnFormat = 'numeric';
   s.ErrorHandling = 'store';
   s.NullNumberRead = 'NaN';
   s.NullNumberWrite = 'NaN';
   s.NullStringRead = 'null';
   s.NullStringWrite = 'null';
   s.JDBCDataSourceFile = '';
   s.UseRegistryForSources = 'yes';
   s.TempDirForRegistryOutput = '';
   setdbprefs(s)

   % Conecta ao banco de dados.
   conn = database('Mestrado','root','admin');

   % Grava dados do banco de dados.
        fastinsert(conn,'resultadopg',
       {'idAreaAgendaGoiana','idPessoa','idInstituicao','idGrupoCNPq','idRedePesquisa','idRedePesquisaClass'
       ,'qtdeAtribSimm','numrResult'}, x1);
   commit(conn);

   % Fecha conexão com o banco de dados
   close(conn);
end

function mostraResult(x3)
   %O algoritmo gera um relatório das classes similares com pelo menos
   %três atributos em comum
   %
   fprintf('\n');
   fprintf('Resultado da Classificação: \n');
   fprintf('\n');
   fprintf('        Redes similares com pelo menos três atritubos comuns \n');
   fprintf('   area  equipe  instituicao grupo   rede    rede      atributos \n');
   fprintf('                  parceira    cnpq    similar classe    comuns \n');
   fprintf('   ------------------------------------------------------------------\n');
   [n,m]=size(x3);
   for i=1:n
      for j=1:m-1
          r=x3(i,j);
          fprintf('%8d ',r);
      end
      fprintf('\n');
   end
end

function [redes,areas,inst,grupo]=resultBD
   % Busca descriçao dos resultados na base de dados para o
   % Relatório final.
```

```
    % Seta preferencias com setdbprefs.
    s.DataReturnFormat = 'cellarray';
    s.ErrorHandling = 'store';
    s.NullNumberRead = 'NaN';
    s.NullNumberWrite = 'NaN';
    s.NullStringRead = 'null';
    s.NullStringWrite = 'null';
    s.JDBCDataSourceFile = '';
    s.UseRegistryForSources = 'yes';
    s.TempDirForRegistryOutput = '';
    setdbprefs(s)

    % Estabelece conexão com banco de dados, usando driver ODBC.
    conn = database('Mestrado','root','admin');

    % Le dados do banco de dados.
    parm=num2str(numrResult);

    %Busca descriçao das Redes de Pesquisa
    result = exec(conn,['Select distinct r.idRedePesquisa, q.Titulo FROM resultadopg r, redespesquisa q where r.idRedePesquisa=q.idRedePesquisa and  r.`qtdeAtribSimm` > 2 and r.`numrResult`= ',parm]);
    result = fetch(result);
    redes = result.Data;

    %Busca descriçao das Áreas da Agenda Goiana
    result = exec(conn,['Select distinct r.idAreaAgendaGoiana, a.NomeAreaAgendaGoiana FROM resultadopg r, areasagendagoiana a where r.idAreaAgendaGoiana=a.idAreaAgendaGoiana and r.`qtdeAtribSimm` > 2 and r.`numrResult`=',parm]);
    result = fetch(result);
    areas = result.Data;

    %Busca descriçao das Instituições Parceiras
    result = exec(conn,['Select distinct r.idInstituicao, i.NomeInstituicao FROM resultadopg r, instituicoes i where r.idInstituicao = i.idInstituicao and r.`qtdeAtribSimm` > 2 and r.`numrResult`=',parm]);
    result = fetch(result);
    inst = result.Data;

    %Busca descriçao dos Grupos do CNPq
    result = exec(conn,['Select distinct r.`idGrupoCNPq`, g.nomeGrupoCNPq FROM resultadopg r, grupos_cnpq g where r.idGrupoCNPq = g.idGrupoCNPq and r.`qtdeAtribSimm` > 2 and r.`numrResult`=',parm]);
    result = fetch(result);
    grupo = result.Data;

    % Fecha conexão com banco de dados.
    close(result)
    close(conn);
  end

  function mostraRelFinal(redes,areas,inst,grupo)
    %Mostra a descrição dos atributos comuns relacionados às classes
    %
    fprintf('\n');
    fprintf('  Descrição dos atributos comuns relacionados às Redes: %d %d %d  \n',rede1, rede2, rede3);
    fprintf('  idArea  Area da Agenda Goiana \n');
    fprintf('  ----------------------------------------------------------------------------------\n');
    [n,m]=size(areas);
    for i=1:n
      for j=1:m
        if iscellstr(areas(i,j))==1
```

```
            c=char(areas(i,j));
            fprintf('%8s ',c);
          else
            c=areas{i,j};
            fprintf('%8d  ',c);
          end
        end
        fprintf('\n');
      end

      fprintf('\n');
      fprintf('  idInst  Instituições Parceiras \n');
      fprintf('  ----------------------------------------------------------------------------------------\n');
      [n,m]=size(inst);
      for i=1:n
        for j=1:m
          if iscellstr(inst(i,j))==1
            c=char(inst(i,j));
            fprintf('%8s ',c);
          else
            c=inst{i,j};
            fprintf('%8d  ',c);
          end
        end
        fprintf('\n');
      end

      fprintf('\n');
      fprintf('  idGrupo  Grupos de Pesquisa do CNPq \n');
      fprintf('  ----------------------------------------------------------------------------------------\n');
      [n,m]=size(grupo);
      for i=1:n
        for j=1:m
          if iscellstr(grupo(i,j))==1
            c=char(grupo(i,j));
            fprintf('%8s ',c);
          else
            c=grupo{i,j};
            fprintf('%8d  ',c);
          end
        end
        fprintf('\n');
      end

      fprintf('\n');
      fprintf('  Descrição das Redes de Pesquisa com indicativo de similaridade às Redes: %d %d %d  \n',rede1,
rede2, rede3);
      fprintf('  idRede  Titulo Rede \n');
      fprintf('  ----------------------------------------------------------------------------------------\n');
      [n,m]=size(redes);
      for i=1:n
        for j=1:m
          if iscellstr(redes(i,j))==1
            r=char(redes(i,j));
            fprintf('%8s ',r);
          else
            r=redes{i,j};
            fprintf('%8d  ',r);
          end
        end
```

```
        fprintf('\n');
      end
   end

%----------------------------Corpo do Programa -----------------------------
format short
fprintf('Processando Classificação... \n');
fprintf('-----------------------------\n');
%1.O conjunto de dados de testes é inicializado através de acesso direto
%ao banco de dados, por meio do comando select da linguagem SQL
  [X]=leBD;
  [nX,mX] = size(X);
  t=mX+1; a2=0; x1=[];
%2.Executar a classificação dos dados de teste(X)
  for a = 1:nX
      x = X(a,1:mX-1);
      %Avalia a expressão da árvore
      S=solvTree(y,x);
      %Verifica o caso em que nenhuma classe ou a mesma classe é encontrada,
      %eliminando estes casos do resultado final
      if S>0
        if X(a,mX)~=S
          a2=a2+1;
          for a1=1:mX
             x1(a2,a1)=X(a,a1);
          end
          x1(a2,t)=S;
        end
      end
   end

%3.          Analisar a similaridade entre a classe original e a nova classe
%encontrada pelo classificador.
   if isempty(x1)==1
      disp ('Não foram encontradas redes similares')
   else
      %3.1Mapear os valores dos atributos preditivos de(X) que pertencem a(D)
      [x2]=mapeamento(x1);

      %3.2Verificar a quantidade de atributos preditivos mapeados e criar
      %uma nova coluna em (X) com a quantidade de atributos encontrados
      [x3]=qtdeAtribMap(x1,x2,mX);

      if isempty(x3)==1
         disp ('Não foram encontradas redes similares com pelo menos três atritubos comuns')
      else
         %4.Gravar resultado do classificador no banco de dados
         gravaResultBD(x3);

         %5.Mostrar resultado do classificador
         mostraResult(x3);

         %5.1Busca no banco de dados a descriçao do resultado
         [redes,areas,inst,grupo]=resultBD;

         %5.2Mostra estatísticas do resultado
         mostraRelFinal(redes,areas,inst,grupo);
      end
   end
end
```