UNIVERSIDADE FEDERAL DA PARAÍBA

Centro de Ciências Exatas e da Natureza

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Mariana de Luna Freire Duarte

MINERAÇÃO DE DADOS USANDO PROGRAMAÇÃO GENÉTICA

João Pessoa – PB

Agosto / 2012

Mariana de Luna Freire Duarte

MINERAÇÃO DE DADOS USANDO PROGRAMAÇÃO GENÉTICA

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática do Departamento de Informática da Universidade Federal da Paraíba como requisito parcial para obtenção do título de Mestre em Informática

Orientador: Valéria Gonçalves Soares.

Co-Orientador: Lucídio dos Anjos Formiga Cabral.

João Pessoa – PB

Agosto / 2012

D812m Duarte, Mariana de Luna Freire.

Mineração de dados usando programação genética / Mariana de Luna Freire Duarte.- João Pessoa, 2012.

79f. : il.

Orientadora: Valéria Gonçalves Soares

Co-orientador: Lucídio dos Anjos Formiga Cabral

Dissertação (Mestrado) – UFPB/CCEN

1. Informática. 2. Programação Genética. 3. Banco de Dados Geográficos. 4. Mineração de Dados Espaciais. 5. Algoritmos Evolucionários.

UFPB/BC CDU: 004(043)

Mariana de Luna Freire Duarte

MINERAÇÃO DE DADOS ESPACIAIS USANDO PROGRAMAÇÃO GENÉTICA

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática do Departamento de Informática da Universidade Federal da Paraíba como requisito parcial para obtenção do título de Mestre em Informática

Área de concentração: Computação distribuída.
Orientador (a): Valéria Gonçalves Soares.
Co-Orientador: Lucídio dos Anjos Formiga Cabral.

COMISSÃO EXAMINADORA

_____

Prof. Dra. Valéria Gonçalves Soares

(Orientador – DI-UFPB)

_____

Prof. Dr. Lucídio dos Anjos Formiga Cabral

(Co-Orientador – DI-UFPB)

_____

Prof. Dr. Clauirton de Albuquerque Siebra

Examinador Interno (DI-UFPB)

_____

Prof. Dr. Luiz Satoru Ochi

Examinador Externo (UFF)

João Pessoa, ___ de _____ de 2012.

Agradecimentos

Agradeço a Deus primeiramente, por me dar o dom da vida, me conceder saúde e sabedoria necessária para caminhar durante toda a minha vida. Agradeço também por se fazer presente ao meu lado guiando os meus passos. E a Maria, por me cobrir com Seu Manto Sagrado, protegendo-me de qualquer mal.

Aos meus pais, Jorge e Rosângela, por dedicar os seus dias a mim e as minhas irmãs, batalhando para nos oferecer sempre o melhor. Obrigada pelo carinho, amor, compreensão, conselhos e críticas que me trouxe amadurecimento.

Às minhas irmãs, Manuela e Marcela, pela compreensão e carinho, contribuindo da melhor maneira para a realização desta pesquisa. Muito obrigada.

Ao meu namorado, Glauciano Almeida, que com todo o seu carinho, amor e paciência me auxiliou em todos os momentos na realização desta pesquisa.

A todos os meus familiares, tios/tias, primos/primas que de alguma forma contribuíram para a realização deste trabalho.

As minhas amigas Jaiana Lopes, Camila César, Carolina Oliveira, e amigo Yuri Bandin, que sempre estiveram ao meu lado, trazendo alegria e distração. E aos demais que direta ou indiretamente cooperaram de alguma maneira.

A minha orientadora, Valéria Gonçalves, e co-orientador, Lucídio Cabral, pela dedicação, compreensão e conhecimentos transmitidos para a realização deste trabalho.

“Pedi e se vos dará. Buscai e achareis. Batei e vos será aberto. Porque todo aquele que pede, recebe. Quem busca acha. A quem bate, abrir-se-á” (Mt 7, 7-8)

RESUMO

A mineração de dados tornou-se uma importante atividade para o processo de tomada de decisão para grandes ou pequenas corporações, pois a partir dela é possível extrair informações relevantes e não triviais de forma que correções e ajustes em estratégias econômicas e administrativas possam ser selecionadas. Assim, vê-se um aumento no armazenamento de dados geográficos, de tal maneira que a mineração de dados convencionais não suporta realizar a extração de conhecimento em um banco de dados de elevada dimensão. De acordo com a literatura atual, poucas ferramentas capazes de extrair conhecimento a partir de dados geográficos são encontradas, principalmente, quando a base de dados é composta por dados convencionais (numéricos e textuais) e geográficos (ponto, linha e polígono). Este trabalho tem como objetivo principal apresentar um novo algoritmo, chamado DMGP, para a atividade de mineração de dados espaciais utilizando os dois tipos de dados para realizar a extração de informações de uma determinada base. O algoritmo em questão tem como base o algoritmo DMGeo que, por sua vez, também visa extrair conhecimento a partir dos dois tipos de dados. Estes algoritmos são baseados na Programação Genética e foram desenvolvidos a fim de obter regras de classificação de padrões existentes nos atributos numéricos e geográficos. Visando obter um melhor desempenho para o DMGeo, foi proposto a utilização das meta-heuríticas GRASP e ILS no funcionamento do algoritmo DMGP para aperfeiçoar os indivíduos das populações geradas. Tais meta-heurísticas foram usadas para gerar a população incial e para realizar uma perturbação de alguns indivíduos, com o intuito de encontrar soluções melhores.

Palavras-chave: Banco de dados Geográficos, Mineração de Dados Espaciais, Algoritmos Evolucionários, Programação Genética.

Abstract

Data mining has become an important activity for decision-making in large and small companies since it allows the extraction of relevant and non-trivial information so that corrections and adjustment in administrative and economic strategies could be selected. Consequently, an increase in the geographical data storage is seen in such a way that conventional data mining cannot carry out the extraction of knowledge from a high dimension database. According to the current literature, there are few tools capable of extracting knowledge from geographical data, mainly if the database is made of conventional (numeral and textual) and geographical (point, line and polygon) data. The aim of this study is to present a new algorithm for spatial data mining – DMGP using the two types of data to carry out the information extraction from a determined base. This algorithm is based on the DMGeo algorithm which also seeks to extract knowledge from the two types of data. These algorithms are based on Genetic Programming and were developed to obtain classification rules of patterns existing in the numeral and geographical attributes. To obtain a better performance for the DMGeo, the use of meta-heuristic GRASP and ILS in the performance of DMGP algorithm was proposed to improve the individuals from the generated population . GRASP and ILS were used to generate the initial population and disturb some individuals aiming at finding better solutions.

Keywords: Geographic Databases, Spatial Data Mining, Evolutionary Algorithms, Genetic Programming.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS / ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AE –Algoritmos Evolucionários.

AG - Algoritmos Genéticos.

BDG - Bancos de Dados Geográficos.

DMGeo - *Niched Genetic Programming Algorithm for Geographic Data Mining*.

GRASP – *Greedy Randomized Adaptative Search Procedure* - Procedimento de Busca Gulosa Adaptativa Aleatória.

ILS - *Iterated Local Search* – Busca Local Iterativa.

KDD –*Knowledge Discovery in Databases*-Descoberta do Conhecimento em Banco de Dados.

LCR - Lista de Candidatos Restrita.

NGAE - *Niched Genetic Algorithm with Elitism*.

DCBG - Descoberta de Conhecimento em Banco de Dados Geográficos.

PG - Programação Genética.

SIG - Sistemas de Informação Geográfica.

SGBD - Sistemas Gerenciados de Banco de Dados.

SGBDGeo - Sistemas Gerenciados de Banco de Dados Geográficos.

SQL *Structured Query Language*.

SUMÁRIO

CAPÍTULO 1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Nos dias atuais várias áreas de conhecimento utilizam uma grande quantidade de dados geográficos, dentre elas podemos citar: sistema de sensoriamento remoto, de transporte, de telecomunicações, etc. E estes dados ficam armazenados em Sistemas Gerenciados de Banco de Dados Geográficos (SGBDGeo), através dos quais, podem ser manipulados por Sistemas de Informação Geográfica (SIG). Todavia, tais sistemas não possuem a capacidade de gerar um conhecimento sobre os dados a partir de novas informações, que a priori não eram conhecidas pelos usuários. Dessa forma, para obter tal conhecimento faz-se necessário a utilização de técnicas de Descoberta do Conhecimento em Banco de Dados (KDD) [Pedrosa, 2010].

O KDD é um processo complexo que tem o objetivo de identificar relacionamentos e informações úteis, que estão implícitas no repositório de dados. A área de estudo responsável por isto é a Mineração de Dados (*Data Mining*) e está relacionada ao desenvolvimento de métodos e técnicas para extrair algum conhecimento significativo, porém implícito em base de dados.

A descoberta de conhecimento em base de dados geográficos, um tipo particular do KDD, está ligada a descoberta de padrões espaciais interessantes, como por exemplo, os relacionamentos entre dados espaciais e não-espaciais, bem como as restrições entre os objetos geográficos e outras características que não estão explicitamente armazenadas no banco. Para isto é preciso que as técnicas de mineração sejam adequadas a extrair conhecimento a partir destes dados geográficos.

Para armazenar, manipular e analisar dados geográficos existem sistemas computacionais, chamados de Sistema de Informações Geográficas (SIG). Os dados geográficos são denominados objetos ou fenômenos do mundo real em que a localização geográfica é uma característica indispensável para tratá-los. Como o volume de dados é normalmente elevado há dificuldades em analisar o comportamento destes tipos de dados. Assim, vêm surgindo diversas formas para extrair um conhecimento não trivial dos grandes e

diferentes repositórios de dados, fazendo com que a mineração de dados, se torne uma área de pesquisa promissora e em amplo crescimento.

Uma das maiores dificuldades em extrair tal tipo de conhecimento provém dos relacionamentos existentes entre os tipos de dados geográficos, pois diferentemente dos dados convencionais, estes dados são dependentes, ou seja, para analisar uma determinada propriedade (atributo) é necessário também analisar toda a sua vizinhança, buscando todos os relacionamentos existentes entre o atributo estudado e os que estão ao seu redor, seja ele numérico, textual ou geográfico. Nos dias atuais, as ferramentas não possuem a capacidade de extrair uma informação levando em consideração atributos geográficos e convencionais ao mesmo tempo, pois os algoritmos de mineração de dados que compõe tais ferramentas não foram desenvolvidos para analisar dados geográficos. Neste contexto, é preciso realizar um pré-processamento a fim de transformá-los em dados convencionais, de tal maneira que seja possível utilizar esses algoritmos de mineração.

Existem na literatura diversas ferramentas e técnicas que auxiliam a atividade de extração do conhecimento a partir de bases de dados, tais como árvores de decisão, redes neurais artificiais e ferramentas evolucionárias [Pereira, 2010]. As ferramentas evolucionárias se destacam por possuírem características como simplicidade, robustez e eficiência.

Estas ferramentas evolucionárias usam mecanismos baseados na Teoria de Evolução das Espécies, proposto por Darwin [Darwin, 2003], onde o autor afirma que "As espécies evoluem pelo princípio da seleção natural e sobrevivência do mais apto.", e nos princípios de herança genética, descritos por Mendel. Dentre os diferentes grupos de algoritmos evolucionários existentes na literatura, destacam-se os Algoritmos Genéticos (AG) e a Programação Genética (PG). Nos AGs, uma população de soluções de um problema é evoluída através de operadores ao longo de várias iterações. As técnicas de mutação, seleção natural, recombinação (*crossover*) são usadas por estes algoritmos para evoluir as populações. Já a programação Genética é um método de geração automática de programas de computador, baseada na simulação de indivíduos, com o intuito de realizar uma tarefa específica solucionando assim um determinado problema.

No contexto da Mineração de Dados Espaciais, para se obter uma ferramenta eficiente de mineração de dados espaciais é necessário que a mesma manipule todos os dados a fim de buscar correlações existentes entre eles, independente de que tipo seja, o que atualmente não

existe. Assim as ferramentas evolucionárias vêm ganhando espaço na área de mineração de dados por extrair uma informação mais precisa da base em estudo. Neste contexto, a principal meta deste trabalho é apresentar um algoritmo capaz de realizar de forma eficiente a extração de informações de dados convencionais e geográficos ao mesmo tempo, assim como fez [Pereira, 2010]. O algoritmo foi desenvolvido especificamente para a atividade de classificação de dados e foi denominado como *Data Minig Genetic Programming* – DMGP. Este utiliza a programação genética, para determinar a melhor regra de classificação para a extração de conhecimento útil e interessante a partir dos dados convencionais e geográficos, analisados simultaneamente. Tem como diferencial do algoritmo DMGeo [Pereira, 2010], o uso de metaheurísticas, chamadas GRASP e ILS, usadas para gerar a população inicial e para melhorar de um determinado indivíduo através de um mecanismo de Busca Local, respectivamente.

1.2 OBJETIVOS

Este trabalho tem por objetivo apresentar uma nova forma para extração de conhecimento útil e interessante a partir de dados convencionais e geográficos baseado na atividade de classificação da Mineração de Dados Espaciais. Com base nisso o proposto é desenvolver um novo algoritmo, usando um tipo de algoritmo evolucionário, a Programação Genética, e meta-heurísticas em algumas etapas do algoritmo, a geração da população inicial e perturbação de um indivíduo, a fim de gerar regras de classificação mais precisas, para a atividade de classificação da Mineração de Dados Espaciais. Sendo este o diferencial do algoritmo DMGeo, em que a geração da população inicial é completamente aleatória e não usa nenhum mecanismo ou meta-heurística para perturbar algum indivíduo da população.

Para atingir tal objetivo faz-se necessário conhecer as ferramentas e métodos existentes para a mineração de dados geográficos e aplicar tais métodos desenvolvidos em problemas reais a fim de validá-los.

1.2.1 Objetivos Específicos

Com a finalidade de se tentar atender o objetivo geral descrito anteriormente, foram colocados os seguintes objetivos específicos:

- Revisão Bibliográfica de Mineração de Dados Espaciais e Programação Genética;
- Geração da população inicial através da metaheurística GRASP;
- Inclusão de um mecanismo de Busca Local usando a metaheurística ILS;
- Estudo da influência dos parâmetros de Mutação e *Crossover* sobre o algoritmo desenvolvido;
- Comparação dos resultados com outros trabalhos existentes na literatura.

1.3 ORGANIZAÇÃO

Este documento está organizado em cinco capítulos. No capítulo 2, são mostrados os conceitos sobre Banco de Dados Geográficos, Mineração de dados e Mineração de dados espaciais, Algoritmos Evolucionários, incluindo Algoritmos Genéticos e Programação Genética.

O capítulo 3 apresenta uma revisão bibliográfica, destacando os principais trabalhos que buscam realizar a mineração de dados convencionais e geográficos. Mostrando também as ferramentas evolucionárias usadas para tal atividade.

No capítulo 4, apresenta-se o algoritmo proposto neste trabalho, baseado em programação genética voltado para mineração de dados geográficos, bem como os resultados computacionais gerados pelo mesmo.

E por fim, no capítulo 5, são expostas as considerações finais do presente trabalho e trabalhos futuros.

CAPÍTULO 2 FUNDAMENTAÇÃO TEÓRICA

Os bancos de dados geográficos (BDG) são desenvolvidos para armazenar dados com características geográficas, ou seja, dados que estão associados a alguma localização geográfica. Conforme [Cavalcanti, 2005] estes bancos de dados manipulam os dados através da análise espacial das coordenadas geográficas e do relacionamento espacial entre elas. Pode-se definir uma entidade geográfica como um fenômeno ou um objeto do mundo real que apresenta atributos associados à sua localização na superfície terrestre, em um determinado período de tempo [Pedrosa, 2010].

Os dados geográficos (ou espaciais) são caracterizados por quatro componentes: características não-espaciais, que representam atributos textuais do fenômeno, como o nome e os atributos específicos dos mesmos; características espaciais, que determinam a posição na superfície terrestre; características temporais, que expressam as variações de tempo de um determinado dado e, por fim, características gráficas, que são as representação pictóricas destes dados [Pedrosa, 2010].

Pode-se afirmar que os Banco de Dados Geográficos são coleções de dados georreferenciados, manipulados através de um Sistema de Informação Geográfica (SIG). De acordo com [Pedrosa, 2010] os SIG's são sistemas que possibilitam a manipulação, a análise e o armazenamento dos dados geográficos. Esses diferem dos demais sistemas de informação por possuir a capacidade de armazenar tanto os atributos descritivos (atributos não-espaciais) como as geometrias dos diferentes tipos de dados geográficos [Câmara, 2005].

Segundo [Câmara, 2005], os dados geográficos possuem modelos de representação. Os dois principais são: modelo de geo-campos e o modelo de geo-objetos. O autor afirma que "o modelo geo-campos enxerga o espaço geográfico como uma superfície contínua, sobre a qual variam os fenômenos a serem observados". Como exemplo tem-se os mapas de vegetação ou de temperatura de uma região. Já em relação ao modelo geo-objetos o autor declara que "o modelo geo-objetos, representa o espaço geográfico como uma coleção de entidades distintas e identificáveis, onde cada entidade é definida como uma fronteira fechada". Estes modelos são mapeados para estrutura de dados matriciais e vetoriais.

Na representação vetorial as informações sobre pontos, linhas e polígonos são codificadas através de uma coleção de coordenadas (x, y). Este tipo de representação é simples, frequentemente utilizada e constituída por entidades espaciais básicas, especializadas

em atributos e localização geográfica. Dessa forma, pode-se afirmar que esta representação não é indicada para representar características continuamente variantes, como a temperatura de uma região por exemplo. Já a estrutura matricial (*raster*) é representada como uma matriz *m x n*, onde *m* é o número de colunas e *n* o número de linhas. Cada célula corresponde a um valor de um determinado atributo, e é individualmente acessada por suas coordenadas. Segundo [Pereira, 2010] a estrutura matricial descreve o espaço como uma superfície plana, em que cada célula da matriz representará um pedaço do terreno. Diferente da vetorial ela armazena características ou fenômenos que variam continuamente com o tempo.

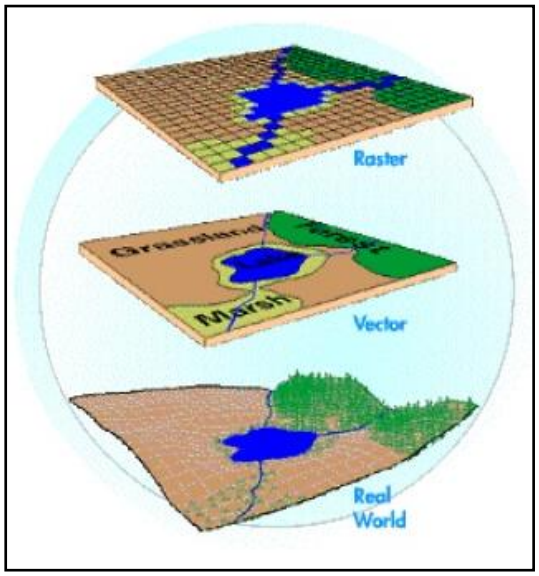

Figura 2.1: Representação Matricial e Vetorial.
Fonte: Pedrosa, K. A. GeoMiningVisualQL: Uma linguagem de consulta visual para Mineração de Dados Geográficos. Dissertação de Mestrado, João Pessoa, 2010.

2.1 TIPOS DE RELACIONAMENTOS ESPACIAIS (RELAÇÕES ESPACIAIS)

Há basicamente três tipos relacionamentos espaciais que são importantes para o armazenamento e recuperação da informação, pois fornece semântica e consistência geométrica às análises realizadas sobre os objetos geográficos, a saber: topológicos, direcionais ou métricos.

Dentre estes relacionamentos, o mais usado são os relacionamentos topológicos, que podem ser definidos pelo tipo de intersecção que ocorre entre duas entidades espaciais. Por

exemplo, é interessante conhecer quais são as cidades vizinhas que fazem fronteira com uma grande cidade ou ainda quais ruas cruzam uma avenida importante.

Os relacionamentos topológicos são definidos com base em modelos de intersecções propostos por [*Egenhofer*e *Herring*, 1991], o primeiro deles foi o modelo de 4-intersecções, que descreve as relações topológicas binárias em termos das intersecções dos seus interiores e limites entre dois objetos. Este relacionamento considera oito relações topológicas binárias: cruza, contém, dentro, cobre, é coberto por, igual, disjunto e sobrepõe. Porém, este modelo não suporta a definição de relacionamentos entre entidades mais complexas. Dessa forma foi proposto o modelo de 9-intersecções (Figura 2.2), que além de descrever as intersecções dos seus interiores e limites, considera também as intersecções dos exteriores entre dois objetos, ou seja, considera os resultados das interseções entre fronteiras, interiores e exteriores de duas geometrias.

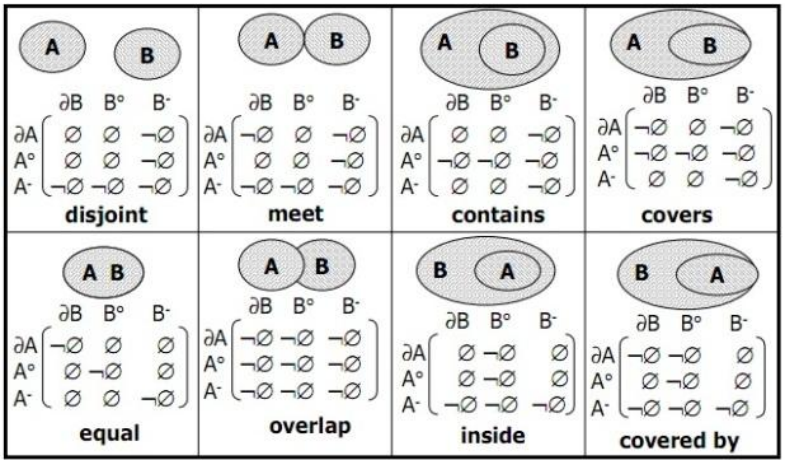

Figura 2.2: Matriz de 9-Interseções para relações entre duas regiões.

Fonte:Câmara, G. Banco de Dados Geográficos, EditoraMundoGEO, Curitiba, 2005.

2.2 DESCOBERTA DE CONHECIMENTO EM BANCO DE DADOS (KDD)

Segundo [Bogorny, 2006] a quantidade de dados de vários formatos e fontes já se tornou tão extensa que um ser humano não é capaz de analisá-los sozinho. Para isto, existem diversas técnicas e ferramentas para automatizar este processo de análise e descoberta de conhecimento.

O processo de Descoberta do Conhecimento (*Knowlegde Discovery in Databases*– KDD) tem por objetivo extrair padrões de dados que sejam válidos, novos, potencialmente úteis e compreensíveis, propondo melhorar o entendimento de um problema. Este processo é interativo, pois o usuário pode intervir e controlar o curso das atividades, e iterativo, por possui uma sequência finita de passos onde os resultados de cada uma deles dependem do resultado dos que os precedem.

O processo contém vários passos para auxiliar o analista do domínio (especialista do domínio dos dados) na tomada de decisão (Figura 2.3), a saber: seleção, pré-processamento, transformação dos dados, mineração dos dados e interpretação e análise.

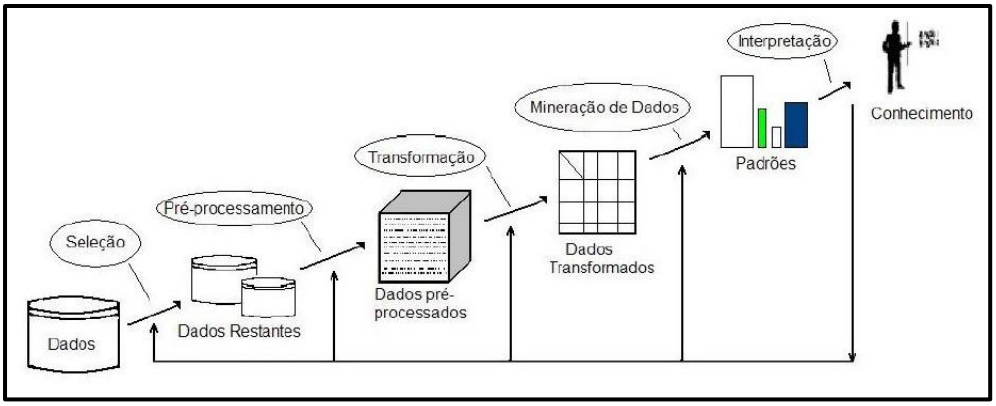

Figura 2.3: Fases do processo de KDD.

Fonte: Castanheira, L. G.; Aplicação de Técnicas de Mineração de Dados em Problemas de Classificação de Padrões. Dissertação de Mestrado. Universidade Federal de Minas Gerais. 2008.

Na fase de **seleção** ocorre a elaboração do conjunto de dados, pertencente ao domínio, contendo os dados que farão parte da análise. Este processo de seleção pode se tornar bastante complexo, pois os dados necessários para análise pode vir de diversas fontes distintas (*data warehouse*, planilhas, sistemas legados), como também podem ter vários formatos.

A fase de **pré-processamento** é bastante árdua, porque a qualidade dos dados irá determinar a eficiência dos algoritmos utilizados na etapa de mineração. Assim, nesta fase devem ser realizadas atividades que removam os dados inconsistentes e redundantes, recuperem dados incompletos e avaliem os possíveis dados discrepantes ao conjunto (*outliers*). É também nessa etapa que são usados métodos para reduzir ou transformar os dados, visando à diminuição dos atributos envolvidos no processo, melhorando assim o desempenho dos algoritmos de análise.

Após serem selecionados, limpos e pré-processados os dados precisam ser armazenados e formatados adequadamente para que os algoritmos de aprendizagem possam ser aplicados. É comum encontrar em grandes empresas computadores rodando em diferentes sistemas operacionais e SGBD distintos, logo estes dados necessitam serem agrupados em um único repositório. Realizado este agrupamento e formatação conclui-se a fase de **transformação de dados**.

A fase de **mineração de dados** é dita a mais importante de todo o processo de KDD, pois é a partir dela que podem ser encontrados os padrões de dados, que por sua vez são classificados como "conhecimento". A etapa de mineração pode ser classificada em: Mineração descritiva, onde são caracterizadas as propriedades gerais dos dados, e Mineração preditiva, onde são executadas inferências nos dados correntes para fazer previsões. Dessa maneira, existem tarefas que compõem esses dois tipos de mineração:

- Tarefas de Descrição: Caracterização e discriminação; Mineração de Padrões Frequentes, Associações e Correlações; Agrupamento (clustering), Outliers;
- Tarefas de Previsão: Classificação e Previsão, Análise Evolutiva;

Por fim, na fase de **interpretação e análise** o conhecimento adquirido na fase de mineração de dados deve ser interpretado e analisado para que o objetivo final seja alcançado. Caso o resultado não seja satisfatório, o processo pode retornar a uma das fases anteriores.

2.2.1 Descoberta de conhecimento em Banco de Dados geográficos (DCBG)

De acordo com [Bogorny, 2006], a descoberta de conhecimento em bases de dados geográficos é um caso particular da descoberta de conhecimento, pois trata-se da exploração a dados georreferenciados, ou seja, dados que possuem referências a objetos geográficos, localizações ou partes de uma divisão territorial. Dessa forma, pode-se concluir que o objetivo deste tipo de descoberta é realizar a extração de padrões espaciais e características interessantes como relacionamentos espaciais e relacionamentos entre dados espaciais e não-espaciais.

Uma característica importante destes bancos de dados é que os dados são interdependentes, isto é, eles estão geograficamente relacionados uns com os outros. Por

exemplo, um estado (Paraíba) está dentro de um país (Brasil). Enquanto nos bancos de dados tradicionais os dados são independentes, sendo esta a principal diferença existente entre estes tipos de banco de dados. [Bogorny, 2006].

Essa dependência dos dados causa dificuldades no processo de descoberta de conhecimento, pois os algoritmos de mineração de dados (onde são encontrados os padrões) foram desenvolvidos para considerar os dados como independentes. Logo, foi preciso estender estas técnicas para que os algoritmos fossem capazes de suportar os dados geográficos.

2.3 MINERAÇÃO DE DADOS

De acordo com [Côrtes, Porcaro, Lifschitz, 2002] a mineração dos dados é a fase de um processo de descoberta do conhecimento com o objetivo de utilizar e analisar dados, por meios automáticos ou semi-automáticos, a fim de descobrir algum padrão ou regra interessante. Ainda conforme os autores citados, esta fase do processo de descoberta de conhecimento torna-se a mais importante dele, pois serão aplicados os algoritmos com o intuito de atingir os objetivos desejados.

As funcionalidades da mineração são utilizadas para especificar os tipos de padrões que serão encontrados nas tarefas de mineração, que são classificadas de duas maneiras: descritiva e preditiva, como definidas anteriormente [Han e Kamber, 2006]. De uma maneira geral, estas duas tarefas de classificação possuem atividades que as compõem. Dentre elas tem-se: classificação, clusterização (agrupamento), regras de associação e *outliers*.

- Classificação

A classificação é definida como um processo para determinar um modelo ou uma função que possa descrever e distinguir classes ou conceitos de dados. Com o objetivo de utilizar este modelo para prever a classe de objeto cuja classe é desconhecida. Neste caso, é realizada uma análise de um conjunto de dados de treinamento para determinar um modelo, que pode ser representado de diversas formas, como regras de classificação do tipo IF-THEN, árvores de decisão, formulação matemática e redes neurais [Han e Kamber, 2006].

- Clusterização (Agrupamento)

Esta técnica pode ser definida como um método que analisa os objetos sem consultar um rótulo conhecido de classe, ao contrário da técnica de classificação e predição, que analisa os objetos rotulados. Dessa maneira, conclui-se que o método possui o objetivo de gerar estes rótulos de classes, baseado em funções métricas (ou de distância). Por exemplo, agrupar países levando em consideração o número da população ou o clima.

Para determinar estes rótulos o método parte do princípio de que não existem rótulos nos dados de treinamento e assim passa a agrupar os dados de acordo com a similaridade dos objetos existentes nos dados. Isto é, se existir uma máxima similaridade entre os objetos eles irão pertencer a uma mesma classe, caso contrário, irão pertencer a classes distintas (Figura 2.4). Cada um destes grupos formados pode ser visto como uma classe de objetos em que é possível derivar regras.

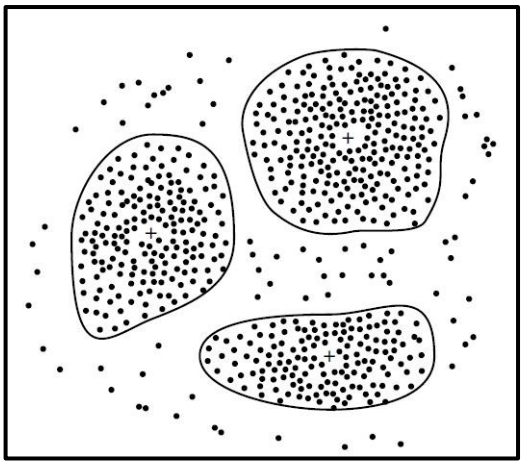

Figura 2.4: Exemplo de clusterização.

Fonte: Han, J. e Kamber, M. Data Mining Concepts and Techniques. Second Edition.Morgan Kaufmann Publishers. 2006.

- Regras de Associação

Conforme [Han e Kamber, 2006] os padrões frequentes são aqueles que ocorrem frequentemente nos dados estudados. Eles podem ser: *itemsets*, sub-sequencias, subestruturas.Os *itemsets* referem-se a um conjunto de itens que frequentemente aparecem juntos em um conjunto de dados tradicionais. As sub-sequencias referem-se, por exemplo, aos padrões que consumidores tendem a comprar. Já as subestruturas podem se referir as diferentes formas estruturais (grafos, árvores, etc.) combinadas com os itens ou sub-

sequencias. Caso a subestrutura ocorra frequentemente ela é considerada uma subestrutura frequente.

Esta mineração de padrões frequentes leva à descoberta de associações interessantes (regras de associação) e correlação entre os dados. Assim, é possível descobrir regras de associação condicionadas a valores de atributos que ocorrem juntos em um determinado conjunto de dados. Uma regra de associação é expressa da forma X → Y. Por exemplo, pode-se determinar quais itens são comprados frequentemente juntos na mesma transação. Para uma loja de eletrônicos, pode-se inferir a seguinte regra:

compra (X, "computador") => compra (X, "software") [suporte = 1%, confiança = 50%]

Ou seja, um cliente que comprar um computador, posteriormente também irá comprar softwares. Nestas regras também pode existir mais de um atributo envolvido.

- *Outliers*

Por definição os *outliers* são objetos que não fazem parte do modelo geral de dados na base de dados, são comumente chamados de pontos fora da curva. Esta técnica tem o objetivo de eliminar estes *outliers* como ruídos e exceções. Em algumas situações específicas detectar este tipo de dados se torna bastante interessante e necessário, como por exemplo, detectar fraude em um sistema bancário.

Esta técnica usa métodos de medidas de distância ou testes estatístico para identificar os objetos que estão muito distantes de qualquer grupo, considerando-os como *outliers*.

2.3.1 Mineração de Dados Espaciais

Segundo [Câmara, Freitas e Neves, 2001] a mineração de dados espaciais é uma extensão da mineração de dados, voltada para domínios de aplicação onde a dimensão espacial é uma característica indispensável no processo de extração de conhecimento.

Com isso nota-se que a mineração de dados espaciais é um processo mais complexo que a tradicional, pois é necessário realizar o tratamento de dados que variam com o tempo e que dependem das relações existentes entre eles e seus vizinhos. Para isto também é preciso aumentar ainda mais a eficiência dos algoritmos que compõe a mineração de dados. Ainda

conforme [Câmara, Freitas e Neves, 2001], a importância da consideração da vizinhança está relacionada ao fato de que para os processos espaciais, as medidas agregadas a um local, tendem a serem semelhante às medidas de locais mais próximos ao mesmo. Logo, pode-se concluir que a relação de vizinhança é a principal diferença entre a mineração de dados e mineração de dados espaciais.

Assim como na mineração de dados tradicional, existem algoritmos que implementam as diferentes técnicas de mineração de dados espaciais, alguns deles foram estendidos da mineração tradicional para suportar os dados espaciais. O maior desafio para a mineração de dados espaciais é a eficiência dos algoritmos utilizados, devido ao volume de dados espaciais, complexidade dos tipos de dados e aos métodos de acesso a estes dados.

As principais técnicas para este tipo de mineração são: generalização, associação espacial, aproximação e agregação, classificação e clusterização [Bogorny, 2006].

- Generalização

Conforme [Bogorny, 2006] a generalização é baseada em hierarquia, partindo de um nível conceitual mais baixo para um nível mais alto, extraindo assim o conhecimento dos dados generalizados. Esta hierarquia pode ser definida pelo usuário ou gerada automaticamente pelo sistema de acordo com análises realizadas nos dados. A autora citada acima afirma que "em bases de dados geográficos, podem ser definidas duas hierarquias: uma, para os dados espaciais e outra, para os dados convencionais.".

Após realizar uma seleção dos dados relevantes, usando uma linguagem de consulta, os dados podem ser inicialmente generalizados de acordo com o seu tipo. Uma generalização de redução é outro tipo de generalização usada, por exemplo, transformar um objeto que é representado espacialmente por uma superfície em um objeto representado por um ponto [Bogorny, 2006].

- Associação espacial

Como especificado anteriormente o objetivo das regras de associação é descobrir itens de uma transação que implicam a presença de outros itens na mesma transação. E possui a forma $X \rightarrow Y$, com suporte e confiança, significando que se um padrão X ocorre em um dado exemplo, então o padrão Y também aparecerá. Os padrões X e Y são denominados antecedente e consequente, respectivamente. A regra de associação espacial também possui o

mesmo objetivo, a diferença entre elas é que na associação espacial é preciso incluir no antecedente ou consequente da regra, uma relação espacial.

Dessa forma pode-se concluir que uma regra de associação espacial descreve uma implicação em um ou mais conjunto de dados sobre um outro conjunto de dados em uma base de dados espaciais. Por exemplo, *is_a(x,house) ^ close_to(x, beach) →is_expensive(x) (90%)*, que implica dizer que 90% das casas localizadas próximas a praias são caras.

Segundo [Bogorny, 2006], quando os objetos estão localizados espacialmente na mesma região é possível identificar relacionamentos espaciais e não-espaciais através das técnicas de classificação e agrupamento. Porém, se os objetos estiverem em regiões geográficas distintas não é possível extrair conhecimento através das técnicas citadas, pois o agrupamento é realizado separadamente para dados espaciais e não-espaciais. "As regras de associação são uma técnica que pode complementar as demais suprindo esta falta, já que os predicados da regra podem ser formados por dados espaciais e não-espaciais" afirma a autora.

Assim como nas regras de associação tradicional, nas espaciais também existem algoritmos para dar suporte a esta atividade, o primeiro deles foi criado por [Koperski e Han, 95] e tem como base uma abordagem de refinamento progressivo, considerando uma etapa intermediária de extração de relações topológicas (*disjoint, intersects, inside, contais,touches, covers, covered-by, equal e crosses)*. Este algoritmo é composto por cinco etapas [Pivato, 2006]:

- **Seleção dos objetos no banco de dados:** ocorre a seleção dos objetos apropriados que serão utilizados para realizar a extração do conhecimento;
- **Busca por relações de proximidade:**ocorre a busca por objetos que apresentem relações de proximidade, a fim de encontrar aqueles que estão mais próximos do objeto de interesse;
- **Filtragem dos resultados de acordo com o suporte:** ocorre a filtragem dos dados a fim de reduzir a quantidade dos mesmos para determinar as relações topológicas;
- **Especialização das relações de proximidade em relações topológicas:** etapa em que são encontradas as relações topológicas especializando as relações de proximidade;
- **Obtenção das regras de associação espacial:** etapa em que ocorre de fato a extração das regras, utilizando o algoritmo *Apriori*.

- Classificação

De acordo com [Bogorny, 2006] a técnica de classificação tem por objetivo enquadrar os dados explorados em classes pré-definidas. Esta técnica permite encontrar regras que dividem o conjunto de objetos em classes, de tal maneira que um objeto seja caracterizado pelo comportamento da classe em que o mesmo está inserido. A classificação dos objetos se dá através das características espaciais ou dos seus atributos descritivos ou ainda através das funções espaciais.

A técnica de classificação espacial possibilita determinar regras para as classes com base nas propriedades não-espaciais e nas relações espaciais que há entre os elementos classificados. Como o processamento das relações espaciais que ocorre entre os objetos pode obter um custo muito elevado, as regras que são encontradas são generalizadas e avaliadas, visando selecionar aquelas que se tornem mais interessante para o estudo em questão [Bogorny, 2006].

- Clusterização

Como é sabido a técnica de clusterização tem como objetivo agrupar os objetos em subconjuntos de acordo com o grau de similaridades entre eles. Estes agrupamentos podem ser baseados em funções de distância. Por exemplo, é possível agrupar casas de uma determinada área, segundo sua categoria, área construída e localização geográfica. Esta técnica é a mais utilizada na mineração de dados, pois possui habilidade de identificar estruturas diretamente dos dados, sem que seja preciso obter um conhecimento prévio dos mesmos.

Os critérios mais comuns usados na clusterização são: homogeneidade e separação. O primeiro refere-se a objetos pertencentes ao mesmo *cluster*, em que os objetos precisam ser os mais semelhantes possíveis. Enquanto o segundo critério, refere-se a objetos de diferentes *clusters*, e eles necessitam ser o mais diferente possível.

De acordo com [Mello, Silva, Souza 2007] “os algoritmos de clusterização podem ser classificados em duas categorias: métodos hierárquicos ou métodos de particionamento.”.

Os **métodos hierárquicos** utilizam as árvores de decisão para dividir os dados da base, decompondo-as recursivamente em conjuntos menores. Esta divisão pode ser realizada de

duas maneiras: *top-down* e *botton-up*, ou podem ser chamadas de divisivos e aglomerativos, respectivamente.

No método *top-down* (ou divisivos), todos os dados selecionados são iniciados no mesmo *cluster*, e durante o decorrer do método o mesmo vai sendo dividido até que seja encontrado apenas um elemento em cada *cluster*. Ao contrário do que acontece no método *botton-up* (ou aglomerativos), todos os dados são elementos únicos, e são agregados até formar um único *cluster*. Estes métodos são denominados hierárquicos porque criam uma relação de hierarquia entre os grupos formados.

Dados *n* objetos para serem agrupados em *k* grupos, os **métodos de particionamento** tem o objetivo de encontrar a melhor partição dos *n* objetos em *k* grupos. Conforme [Bogorny, 2006] "os métodos de particionamento mais utilizados são baseados em um ponto central (média dos atributos dos objetos – *k-means*) ou em um objeto representativo para o *cluster* (*k-medoids*).".

O método *k-means* usa o ponto médio da distância entre os objetos no espaço para representar o centro do grupo. Enquanto o método *k-medoids* usa um objeto espacial que esteja mais próximo do ponto médio da distância euclidiana para representar o centro do grupo [Mello, Silva, Souza 2007].

2.4 ALGORITMOS EVOLUCIONÁRIOS

Diversos métodos de otimização procuram reproduzir algum processo natural, em particular as técnicas de sobrevivência e evolução das espécies. A motivação para este tipo de atividade é que, se determinado mecanismo está presente e é eficaz na natureza, então há expectativas para o seu sucesso quando aplicado a métodos artificiais.

De acordo com [Soares, 2008], os algoritmos evolucionários são baseados no processo evolutivo descrito por Darwin, porém este termo é muito amplo e pode ser empregado aos métodos que encontram soluções a partir de melhorias das soluções das iterações anteriores. Segundo [Talbi, 2009], estes algoritmos são baseados em noções de competição e representam uma classe de algoritmos de otimização iterativa que simula a evolução das espécies. Ainda conforme o autor os algoritmos evolucionários possuem a seguinte forma geral (Figura 2.5).

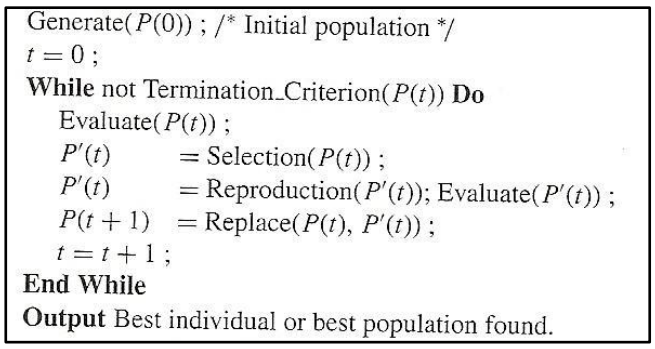

Figura 2.5: Template de Algoritmo Evolucionário.

Fonte: Talbi, E. Metaheuristics from Design to Implementation.Publicadopor John Wiley & Sons.2009.

Inicialmente, a população de indivíduos é gerada randomicamente. Todo indivíduo é codificado para uma possível solução, bem como associado a uma função objetivo (*fitness*). Como se trata de um processo iterativo, em cada etapa do algoritmo os indivíduos que possuem a melhor *fitness* são selecionados. Em seguida, os indivíduos são reproduzidos utilizando os operadores de evolução (mutação, crossover, seleção, etc.). Por fim, o esquema de substituição é aplicado para determinar quais indivíduos da população sobreviverão para a próxima geração. Este processo é realizado até satisfazer o critério de parada. E assim, ao final da iteração, o melhor indivíduo ou a melhor população é retornado.

A teoria de criação de novas espécies e evolução delas inspirou muitos cientistas da computação a criar novos algoritmos evolucionários, a saber: algoritmos genéticos, estratégias evolucionárias e programação evolucionária[Talbi, 2009].

- Algoritmos Genéticos

Conforme [Talbi, 2009] os algoritmos genéticos são considerados como a classe mais popular dos algoritmos evolucionários. Foi criado por J. Holand em 1970. Estes algoritmos geralmente usam uma representação binária, porém já podem ser encontrados com outros tipos de representação.

Ainda de acordo com o autor, estes algoritmos aplicam os operadores de mutação e *crossover*. Inicialmente é usado o operador de *crossover* para determinar duas soluções e

escolher a melhor delas, e em seguida usa-se o operador de mutação para modificar randomicamente o indivíduo promovendo a diversidade na população.

- Estratégias evolucionárias

As estratégias evolucionárias são aplicadas principalmente a problemas de otimização contínua onde as representações são baseadas em vetores de valores reais. Elas usam uma estratégia de substituição elitista, juntamente com o operador de mutação específica normalmente distribuída [Talbi, 2009].

O operador de seleção também é usado nessas estratégias, ele é determinístico e baseado em um ranking de fitness. [Talbi, 2009] afirma que "um indivíduo é composto por uma variável de decisão *float* mais alguns parâmetros de pesquisa". Sua maior vantagem é a eficiência em tempo de complexidade.

- Programação evolucionária

Segundo [Talbi, 2009] a programação evolucionária enfatiza mais o operador de mutação e não utiliza a recombinação. Inicialmente, ela foi usada para solucionar problemas de predição de série temporal ou de aprendizagem de máquina, porém, atualmente pode ser aplicado a problemas de otimização contínua utilizando uma representação de valores reais. O processo de seleção usado na programação evolucionária é o baseado na seleção de torneio estocástico.

- Programação Genética

Uma das pesquisas evolucionárias mais recentes existente nos dias atuais é a Programação Genética (PG). Conforme [Talbi, 2009], a programação genética é definida como uma forma de indução de programas que permite gerar automaticamente programas para solucionar uma dada atividade. Neste algoritmo, o operador de *crossover* é baseado na troca de sub-árvores e o operador de mutação é baseado em trocas aleatórias na árvore. O autor ainda afirma que "um dos principais problemas na programação genética é o crescimento descontrolado de árvores, este fenômeno é chamado de *bloat*.". Usualmente a programação genética é usada nas atividades de aprendizagem de máquina e tarefas de mineração de dados.

2.4.1 ALGORITMOS GENÉTICOS (AG)

Os algoritmos genéticos, desenvolvidos por John Holland em 1970, são uma classe dos algoritmos evolucionários baseados nos princípios da seleção e da genética natural. Todos os métodos computacionais, voltados para este paradigma de seleção e genética natural existentes atualmente são fundamentados nos mecanismos de evolução natural das espécies, em que são criadas novas populações, a partir dos operadores genéticos (seleção, cruzamento e mutação), utilizando indivíduos que são avaliados segundo seu desempenho dentro de um ambiente [Santos, 2008].

Segundo [Dias, 2004], através do processo de seleção natural é possível aumentar a qualidade média da população ao longo do processo evolutivo, obtendo assim indivíduos totalmente adaptados ao ambiente. Isto ocorre devido ao fato da seleção privilegiar os indivíduos que possuem uma alta capacidade de sobrevivência.

O conceito de evolução natural é aplicado à computação para a resolução de problemas computacionais, pois os mecanismos de evolução parecem se adequar a estes problemas nas mais diversas áreas. Os problemas geralmente envolvem busca em espaço muito grande de solução, como por exemplo, extração de um conjunto de regras de classificação a partir de uma base de dados [Santos, 2008].

De acordo com [Santos, 2008], Holland exibe os algoritmos genéticos como uma abstração do processo evolutivo, que permitem importar os conceitos de adaptação, evolução e seleção natural da vida real para o mundo computacional, com o intuito de solucionar problemas que abrangem a busca por uma solução ótima.

A ideia principal de um algoritmo genético é que um indivíduo da população envolvida codifica uma solução candidata para o problema em questão. Geralmente para esta codificação são utilizados vetores de valores binários, inteiros ou reais, que formam o indivíduo, também chamado de cromossomo. E cada um destes indivíduos é avaliado por uma função de *fitness* (ou função objetivo), onde será julgada a qualidade de cada indivíduo em relação ao problema modelado. Os indivíduos são evoluídos através da relação entre os melhores indivíduos gerados através dos operadores de seleção (sobrevivência e reprodução do mais apto) e genética natural (cruzamento e mutação).

Em um algoritmo genético uma solução é gerada, e então é testada a sua eficácia na resolução do problema, caso a solução seja adequada para solucionar o problema, obedecendo

às limitações impostas anteriormente, ela é adotada. Caso contrário, a solução é desprezada e o processo recomeça gerando uma nova solução.

O funcionamento de um algoritmo pode ser visualizado na Figura 2.6. Inicialmente o algoritmo gera de forma randômica uma população inicial com *n* indivíduos (cromossomos), que representam possíveis soluções para o problema a ser solucionado. Após selecionados os indivíduos, eles são avaliados por uma função de avaliação, chamada *fitness*. Quanto mais adaptado for o indivíduo melhor será a sua avaliação calculada por essa função. Em seguida, é verificado se a condição de parada foi satisfeita, caso não tenha sido são selecionados *k* indivíduos para o processo de reprodução, através dos operadores genéticos, *crossover* e mutação. Quando uma nova população for gerada todo o processo inicia-se novamente.

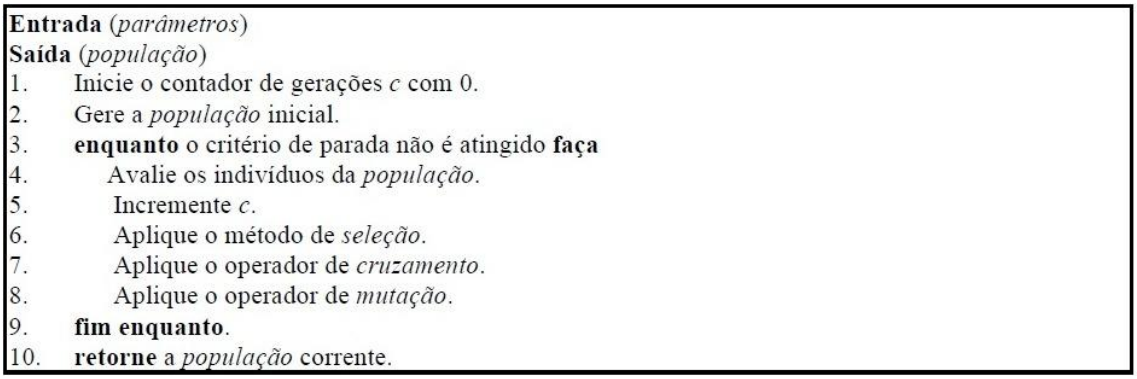
Entrada (*parâmetros*)
Saída (*população*)
1. Inicie o contador de gerações *c* com 0.
2. Gere a *população* inicial.
3. **enquanto** o critério de parada não é atingido **faça**
4. Avalie os indivíduos da *população*.
5. Incremente *c*.
6. Aplique o método de *seleção*.
7. Aplique o operador de *cruzamento*.
8. Aplique o operador de *mutação*.
9. **fim enquanto**.
10. **retorne** a *população* corrente.

Figura 2.6: Estrutura básica de um Algoritmo Genético.

Fonte: Pereira, M. A. Mineração de Dados Espaciais através de Algoritmos Evolucionários. Exame de Qualificação. Universidade Federal de Minas Gerais. 2010.

2.4.2 FUNÇÃODEAVALIAÇÃO

Segundo [Santos, 2008] a função de avaliação, também chamada de *fitness* ou função de aptidão, é uma forma encontrada pelos algoritmos genéticos para especificar a qualidade de um indivíduo dentro da solução do problema. Os algoritmos genéticos podem ser utilizados em vários problemas de otimização, pois usa a função de avaliação para computar a aptidão de um indivíduo de uma população durante o seu processo evolutivo [Dias, 2004].

A busca que é realizada no algoritmo genético é controlada pela função de avaliação. Assim, cada indivíduo que compõe a solução é avaliado e recebe uma nota. Esta é usada para

determinar quais indivíduos são aptos a reprodução, essa escolha acontece através do método de seleção [Santos, 2008].

2.4.3 SELEÇÃO E REPRODUÇÃO

A cada iteração do algoritmo genético é preciso obter uma nova população sobre a qual serão aplicados os operadores genéticos. Para isto, faz-se necessário a utilização dos métodos de seleção e reprodução dos indivíduos. Ao escolher os indivíduos, com melhor função de avaliação, através do método de seleção, a reprodução é realizada copiando o código genético do indivíduo selecionado para a nova população. Dessa forma, pode-se concluir que o método de seleção corresponde a um mecanismo que permita a sobrevivência dos melhores indivíduos, a fim de compartilhar suas características com as gerações seguintes [Dias, 2004].

Os métodos de seleção usados nos algoritmos genéticos são os mesmo dos algoritmos evolucionários. Os mais utilizados deles são os métodos da roleta e do torneio. No método da roleta, cada indivíduo possui um pedaço da roleta proporcional a sua função de avaliação (como é apresentado na Figura 2.7). E assim cada vez que a roleta é girada um indivíduo é selecionado. A roleta é girada até que seja atingido o número do tamanho da população, podendo um indivíduo ser selecionado diversas vezes. Enquanto no método do torneio, *n* (tamanho do torneio, e seu valor é definido pelo usuário) indivíduos são selecionados aleatoriamente e o mais apto dentre eles é escolhido para reprodução.

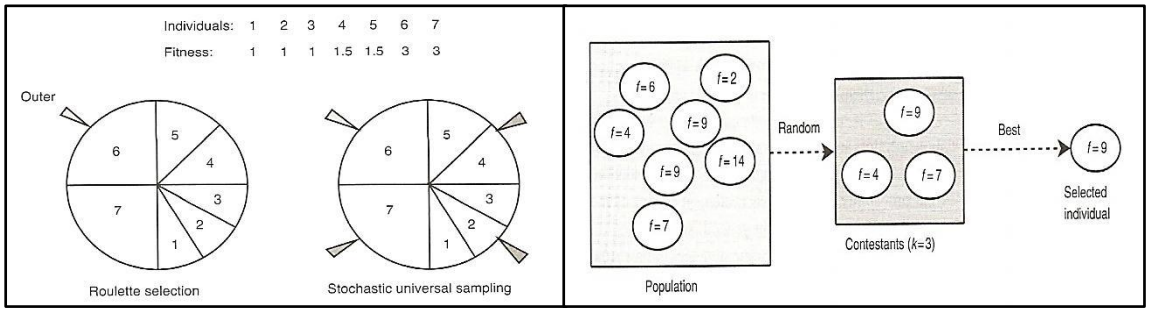

Figura 2.7: Estratégias de seleção: Roleta, Amostragem Universal Estocástica (lado esquerdo) e Torneio (lado direito).

Fonte: Talbi, E. Metaheuristics from Design to Implementation.Publicadopor John Wiley & Sons.2009.

2.4.4 OPERADORES GENÉTICOS

Após a seleção e reprodução dos indivíduos de uma geração, é iniciada uma nova etapa do algoritmo, sobre a qual são aplicados os operadores genéticos. Geralmente os operadores genéticos utilizados em um algoritmo genético tradicional são o operador de cruzamento e o operador de mutação [Dias, 2004].

De acordo com [Goldberg, 1989] o operador de cruzamento tem o objetivo de obter indivíduos melhores a partir dos já selecionados, através da troca de partes de pares dos mesmos. Ou seja, ele possui o intuito de assegurar a troca de material genético entre dois indivíduos (também chamados pais), combinando as informações de ambos a fim de obter melhores indivíduos que os pais [Santos, 2008].

Conforme [Talbi, 2009], há várias representações do operador de cruzamento, a saber:

- Cruzamento de um ponto

Dado dois indivíduos, um ponto de cruzamento é escolhido randomicamente. Após a divisão, os segmentos situados após o ponto de cruzamento são permutados, como pode ser visualizado na Figura 2.8.

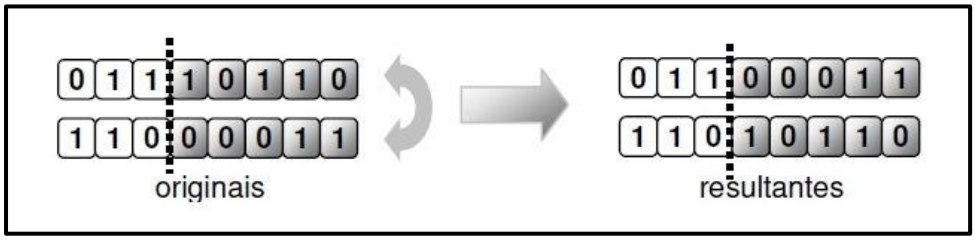

Figura 2.8: Cruzamento de um ponto.

Fonte: Dias, Carlos Rodrigues. Algoritmos Evolutivos para o Problema de Clusterização de Grafos Orientados: Desenvolvimento e Análise Experimental, 2004.

- Cruzamento de dois pontos

Dados dois indivíduos, são escolhidos dois pontos de cruzamento de forma aleatória, e os segmentos formados entre os pontos de cruzamento são trocados (Figura 2.9).

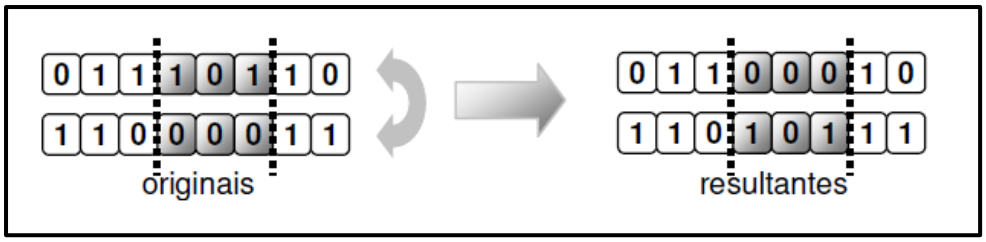

Figura 2.9: Cruzamento de múltiplos pontos.

Fonte: Algoritmos Evolutivos para o Problema de Clusterização de Grafos Orientados: Desenvolvimento e Análise Experimental. Dias, Carlos Rodrigues, 2004.

- Cruzamento uniforme

Nesta forma de cruzamento, inicialmente é gerada uma máscara de dígitos binários escolhidos aleatoriamente, onde o dígito 1 indica que haverá a troca do valor na respectiva posição do indivíduo, equivalente a posição da máscara, e o dígito 0 indica que os valores da respectiva posição devem ser mantidos (Figura 2.10).

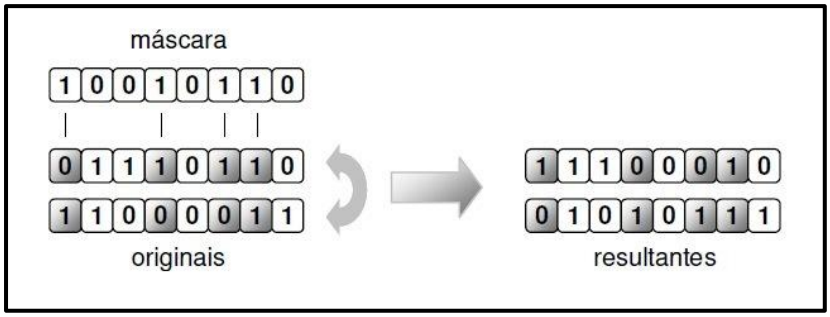

Figura 2.10: Cruzamento Uniforme.

Fonte: Algoritmos Evolutivos para o Problema de Clusterização de Grafos Orientados: Desenvolvimento e Análise Experimental. Dias, Carlos Rodrigues, 2004.

A mutação opera sobre os indivíduos resultantes do processo de cruzamento, diferentemente deste, ela realiza trocas aleatórias de alguns valores dos indivíduos. O operador de mutação tem o propósito de manter a diversidade da população, não permitindo assim que a mesma fique estagnada, e assegurando que serão obtidas novas soluções potenciais. Um exemplo de mutação pode ser visualizado na Figura 2.11, onde um indivíduo está sendo representado por uma cadeia de dígitos binários, em que os valores dos elementos em destaque são permutados.

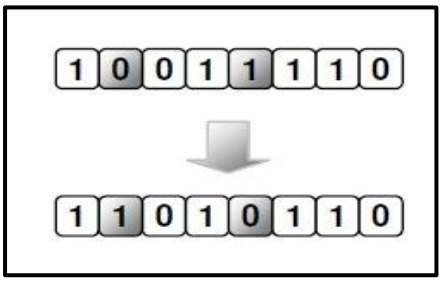

Figura 2.11: Mutação.

Fonte: Dias, C. R.Algoritmos Evolutivos para o Problema de Clusterização de Grafos Orientados: Desenvolvimento e Análise Experimental. 2004.

2.4.5 PROGRAMAÇÃO GENÉTICA (PG)

A programação genética é um método de geração automática de programas de computador, baseada na simulação da evolução de indivíduos, para realizar uma tarefa específica, a fim de solucionar um determinado problema. Para compreender melhor o funcionamento da programação genética, uma questão levantada por [Samuel, 1959] foi apresentada: "Como os computadores podem aprender a resolver problemas, sem serem explicitamente programados para tal?". Assim ao responder a questão levantada conclui-se que a programação genética tem por objetivo alcançar a aprendizagem por indução através da evolução de uma população de programas de computador.

De acordo com [Koza, 1992], um programa de computador para solucionar determinada atividade pode ser construído a partir da combinação de seleção natural darwiniana e operações genéticas. Na Programação Genética, estes programas são representados como árvores de análise, onde os nós representam procedimentos, variáveis, funções ou constantes.

A programação genética pode ser vista como uma especialização do Algoritmo Genético, onde a codificação dos indivíduos passa a ser capaz de representar programas de computador. Dessa forma pode-se afirmar que o mecanismo de programação genética é similar ao do Algoritmo Genético como apresentado na Figura 2.12. A diferença entre eles encontra-se na forma de representação do indivíduo, enquanto que em um algoritmo genético o indivíduo é representado como uma cadeia (binária, real, etc.), na programação genética um indivíduo é codificado na forma de uma árvore.

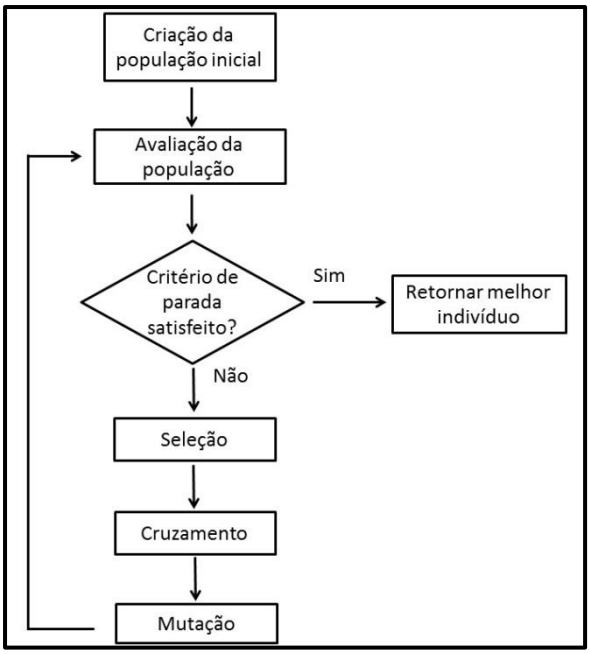

Figura 2.12: Fluxograma de um algoritmo de Programação Genética.

Fonte: Adaptado de Maia, R. D. Programação Genética. Universidade Federal de Minas Gerais. 2007.

Conforme [Koza, 1992] a Programação Genética é iniciada através de uma população inicial, gerada aleatoriamente, formada por funções e propriedades terminais sobre o domínio do problema. Estas funções podem ser operações aritméticas, de programação padrão, matemáticas, lógicas ou de domínio de funções específicas. Após a geração desta população, se dará início à geração de novas populações através dos operadores de evolução. Indivíduos são selecionados para gerar novos filhos, ou ainda para serem substituídos por outros com o objetivo de manter sempre uma boa geração de indivíduos na população. Assim, cada indivíduo que forma a população é analisado de acordo com a sua função de avaliação, assim como ocorre nos algoritmos genéticos.

2.4.5.1 Representação dos Indivíduos

Conforme já mencionado anteriormente, na programação genética os indivíduos são representados através de árvore sintática, possibilitando que os mesmos contenham não apenas valores constantes, mas também funções, isto é, os indivíduos são formados por uma combinação de funções (conjunto de não-terminais) e terminais [Pereira, 2010].

Por exemplo, considere o conjunto de funções como um conjunto de operadores aritméticos e o conjunto dos terminais formado pela variável x e a constante 2, conforme apresentado a seguir [Rodrigues, 2002].

F = {+, -, *, /}

T = {x, 2}

Assim, a partir destes conjuntos podem ser produzidas expressões matemáticas, tais como x*x+2. E sua representação na árvore sintática é apresentada na Figura 2.13.

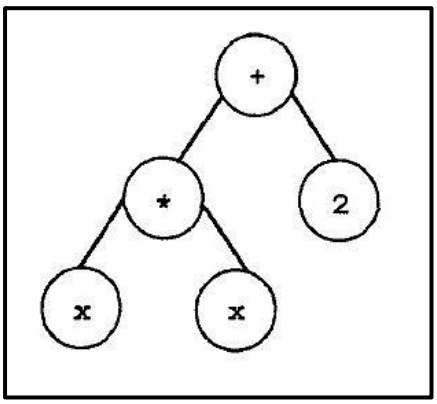

Figura 2.13: Árvore Sintática de x*x+2.

Fonte: Rodrigues, E. L. M. Evolução de Funções em Programação Genética orientada a gramáticas. Dissertação de Mestrado. Universidade Federal do Paraná, Curitiba, 2002.

Segundo [Souza, 2006], o conjunto das funções (ou não-terminais) é composto por operadores aritméticos, funções matemáticas, operadores lógicos, dentre outros. Enquanto o conjunto de terminais é composto por variáveis e constantes.

De acordo com [Pereira, 2010], um indivíduo da Programação Genética deve satisfazer pelo menos duas propriedades, que foram definidas por Koza (1992):

- Suficiência: os conjuntos de funções e terminais devem ser suficientes para representar uma solução para o problema, garantindo assim a convergência do sistema.

- Fechamento: esta propriedade deve ser capaz de garantir que qualquer função do conjunto de funções seja capaz de operar com todos os valores recebidos como entrada, garantindo que as árvores criadas sejam viáveis.

2.4.5.2 População Inicial

Na programação genética o primeiro passo é gerar uma população inicial, que em geral, é criada de maneira aleatória a partir dos conjuntos funções (F) e terminais (T). Inicialmente, determina-se randomicamente uma função $f \in$ F e para cada argumento de F, determina-se um elemento da união $\{F \cup T\}$. Este processo é realizado até que existam apenas terminais nos nós-folhas da árvore. Para que a árvore não se torne muito grande, geralmente, especifica-se um tamanho máximo para a mesma.

Uma característica muito importante que deve ser observada é a "qualidade" da árvore, pois ela é um fator crítico para o processo evolutivo. A população inicial deve conter umsubconjunto significativo do espaço de busca analisado, com o intuito de convergir para uma solução através da recombinação de seus códigos [Rodrigues, 2002].

Como as árvores sintáticas são estruturas complexas, é preciso ter cautela ao gerar a população inicial, para isto existem métodos que auxiliam esta atividade melhorando assim a qualidade da mesma, os mais comuns deles são: *Full*, *Grow*, *half-and-half*(combinação dos métodos *Full*e *Grow*), *random-branch*e *uniform* [Souza, 2006].

- ***Grow*:** Método em que a criação da árvore é realizada de maneira aleatória e a sua profundidade, distância do nó raiz até um determinado nó, é variável, porém a profundidade máxima é respeitada.

- ***Full*:** Ao contrário do método *Grow*, que escolhe aleatoriamente nós do conjunto F e T, o método *Full*, seleciona apenas funções até que um nó terminal seja atingido. Gerando assim árvores completas, todas com a mesma profundidade.

- ***Half-and-Half*:** Combinação dos métodos *Grow*e *Full*, a fim de gerar um número igual de árvores para cada profundidade. O método *Full*é usado 50% das vezes e o método *Grow*nas outras 50%. As desvantagens deste método são:

  - Caso o conjunto de funções seja maior que o conjunto de terminais a tendência será gerar a maior árvore possível;

- A determinação da profundidade máxima é feita de maneira proporcional e não aleatória;

- A faixa de profundidade é fixa (geralmente entre 2 e 6), dependendo do número de argumentos de cada função e independente do tamanho da árvore.

- ***Random-Banch*:** Neste método informa-se o tamanho máximo da árvore e não a sua profundidade. O tamanho máximo é igualmente distribuído entre as árvores de um nó-pai não terminal, gerando muitas árvores que não são válidas.

- ***Uniform*:** Possui o objetivo de garantir que todas as árvores criadas, a partir do conjunto de todas as árvores possíveis, sejam uniformes. O algoritmo é bastante complexo, pois necessita calcular várias vezes o número de árvores possíveis para cada tamanho desejado.

2.4.5.3 Função de Avaliação

A função de aptidão é uma medida usada pela Programação Genética durante a execução do algoritmo para determinar se o programa está apto a se tornar a solução. Ou seja, os melhores programas que resolvem o problema terão a maior chance de serem selecionados para a fase de reprodução, a fim de encontrar a solução [Rodrigues, 2006]. E ela é definida de acordo com o domínio do problema em questão.

De acordo com [Souza, 2006], para que um algoritmo gere uma solução muito próxima da ótima, é preciso que a função de aptidão seja bem definida.

2.4.5.4 Operadores Genéticos

Como já é sabido, para submeter os indivíduos aos operadores genéticos eles necessitam ser devidamente escolhidos para tal atividade. Assim os métodos de seleção, possuem o objetivo de selecionar aqueles indivíduos mais adequados para sofrer alterações, formando assim novas gerações. Para isto é preciso determinar um critério de seleção, o mais usado deles é o valor da aptidão (*fitness*). Logo, aqueles que possuem os melhores valores de aptidão são selecionados.

[Souza, 2006] afirma que "o método de seleção é responsável pela velocidade da evolução e geralmente citado como responsável pelos casos de convergência prematura que

poderão determinar o sucesso do algoritmo evolucionário". Alguns dos métodos de seleção são: **Seleção Proporcional**, que especifica a probabilidade de cada indivíduo para que seja selecionado para a próxima geração; **Truncamento**, a seleção é realizada aleatoriamente entre os T melhores indivíduos, baseado em um valor limiar T, que se encontra em um intervalo de zero a um; **Ranking**, os indivíduos são ordenados de forma crescente de acordo com seu valor de aptidão, a cada um deles é atribuído um número inteiro conforme sua posição no ranking e assim quanto melhor o ranking melhor o valor de sua aptidão e maior chance de ser selecionado;**Torneio**, realizado em sub-conjuntos da população, alguns indivíduos são selecionados aleatoriamente e é realizada uma competição seletiva entre eles [Souza, 2006].

Depois de selecionados os indivíduos, os operadores genéticos são aplicados com o intuito de gerar novas populações. Vários operadores foram criados, porém os mais utilizados e importantes deles são: Reprodução, Cruzamento e Mutação.

- **Reprodução:** um indivíduo da população atual é selecionado e copiado para a próxima geração sem nenhuma alteração em sua estrutura. Seguidamente, este mesmo indivíduo é re-inserido na população, havendo assim duas versões do mesmo na população.

- **Cruzamento:** tem o objetivo de realizar a troca de material genético entre indivíduos pais, ou seja, dois indivíduos são selecionados e ocorre uma combinação de seus materiais genéticos, permutando uma parte de um dos pais por uma parte do outro (Figura 2.14).

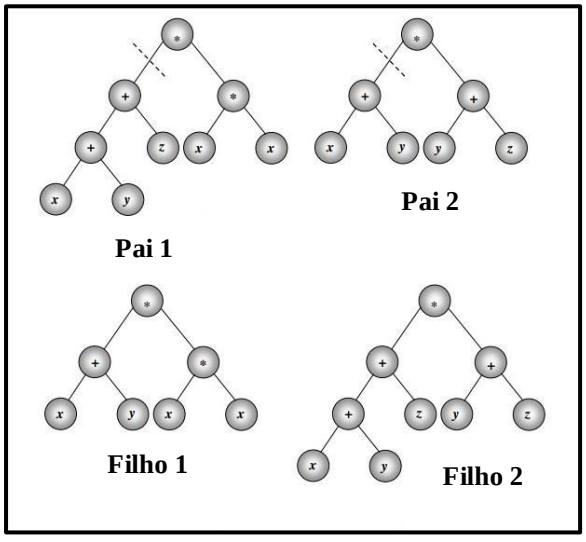

Figura 2.14: Representação do Operador de Cruzamento.

Fonte: Souza, 2006.

- **Mutação:** este operador realiza alterações em apenas um indivíduo. Esta troca é feita selecionando aleatoriamente um ponto do indivíduo e substituindo a sub-árvore pelo nó que foi selecionado por uma nova sub-árvore gerada aleatoriamente. Esta nova sub-árvore formada também está sujeita às mesmas limitações de profundidade e tamanho das árvores geradas na população inicial. Este operador tem o objetivo de diversificar a população, porém é necessário tomar cuidados, pois ao inserir muita diversidade na população a mesma pode não convergir para uma solução ótima. Este procedimento pode ser visualizado na Figura 2.15, em que encontra-se a troca de uma sub-árvore inteira.

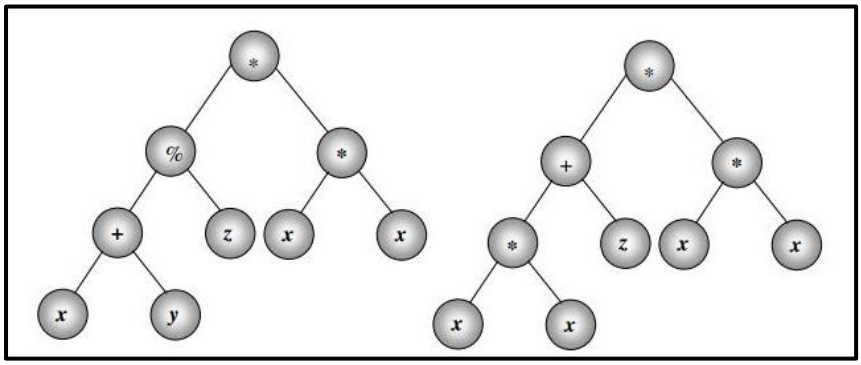

Figura 2.15: Representação do Operador de Mutação.

Fonte: Souza, 2006.

2.5 METAHEURÍSTICAS

As metaheurísticas são procedimentos heurísticos genéricos direcionados prioritariamente para a solução aproximada de problemas computacionalmente difíceis. Elas utilizam diferentes mecanismos para tentar escapar de ótimos locais e aproximar-se o máximo possível de uma solução ótima global. As mais frequentes atualmente são: Busca Tabu, GRASP, ILS, Algoritmos genéticos, VNS, Colônias de formigas.

Para esta pesquisa serão utilizadas as metaheurísticas GRASP e ILS, serão usadas para realizar uma nova forma de reprodução e permutação, respectivamente. Estas serão descritas com mais detalhes nas seções seguintes.

2.5.1 GRASP

De acordo com [Talbi, 2009], a metaheurística GRASP (*Greedy Randomized Adaptative Search Procedure* – Procedimento de Busca Gulosa Adaptativa Aleatória), proposta por Feo e Resende (1995), é uma heurística gulosa iterativa para solucionar problema de otimização. Cada iteração é composta por duas fases: construção e busca local. O objetivo da primeira etapa é construir uma solução viável para o problema. A segunda tem o intuito de fazer uma busca pela vizinhança da solução gerada na fase anterior aprimorando-a até que um ótimo local seja encontrado [Trindade, 2005].

Um pseudo-código da metaheurística GRASP pode ser visualizado no Algoritmo a seguir (Algoritmo 2.1) . Em que pode-se observar que a melhor solução encontrada após a busca local é retornada.

```
1: ler entrada
2: for  k = 1 até número_de_iterações do
3:     solução1 = construção(semente)
4:     solução2 = busca local (solução1)
5:     guarda melhor solução ( solução2 , melhor solução)
6: end for
```

Algoritmo 2.1: Pseudocódigo da Metaheurística GRASP.

Fonte: Peres, F. S. Heurísticas GRASP para o problema de Formação de Célulasde Manufatura. Dissertação de Mestrado. Universidade Federal Fluminense, 2006.

2.5.1.1 Construção

Como já especificado acima nesta primeira etapa da metaheurística cria-se uma solução inicial iterativamente elemento a elemento. Estes elementos que são candidatos a fazer parte a solução são avaliados e a cada um deles é atribuído um valor através de uma função gulosa, que mede o benefício que o elemento escolhido mais recentemente concede à solução já construída, e, assim, ordenados em uma lista chamada LCR (Lista de Candidatos Restrita). Após a inserção dos elementos na lista, são selecionados aleatoriamente aqueles que irão compor a solução parcial. A cada elemento escolhido a lista é atualizada e todos os elementos que ainda a compõe são reavaliados pela função e assim uma nova lista é gerada. Por estas razões o algoritmo é definido como sendo iterativo, guloso, aleatório e adaptativo [Peres, 2006].

Conforme [Trindade, 2005] há duas estratégias para construir a LCR: pelo número de elementos (percentual do tamanho da lista de candidatos) ou pela qualidade de elementos. Na primeira, os elementos candidatos são ordenados de maneira decrescente de benefício de acordo com a função de avalição e os *p* primeiros elementos são incluídos na lista. O valor de *p* é definido como:

$$p = 1 + \alpha(a - 1)$$

Onde a é o número total de candidatos e α é um parâmetro que possui valores definidos no intervalo [0,1]. Pode-se perceber que quando α = 0 o algoritmo torna-se totalmente guloso, escolhendo assim apenas um único elemento a ser inserido na solução. Por outro lado, quando α = 1, o algoritmo torna-se aleatório, logo a lista conterá todos os possíveis candidatos.

Na segunda estratégia, considere um problema de minimização, t um elemento candidato, C um conjunto de todos os elementos candidatos, g(t) como a função gulosa tmin e tmax, o menor e o maior valor encontrado respectivamente e α o fator de aleatoriedade. Assim para incluir um elemento na lista, o mesmo precisa ter o valor da função dentro do intervalo [tmin, (tmin + (1- α) (tmax – tmin)] [Peres, 2006].

O Algoritmo 2.2 ilustra o pseudocódigo da etapa de construção da metaheurística.

```
1: Solução = { }
2: while solução imcompleta do
3:    tmin = min ( g(t) | t ∈ C )
4:    tmax = max ( g(t) | t ∈ C )
5:    LCR = { t ∈ C | g(t) >= ( tmin + ( 1 - α ) ( tmax - tmin) ) }
6:    Escolhe um elemento t ∈ LCR aleatoriamente
7:    Solução = solução ∪ { t }
8: end while
9: return Solução
```

Algoritmo 2.2: Pseudocódigo etapa de Construção Gulosa.

Fonte: Peres, F. S. Heurísticas GRASP para o problema de Formação de Células de Manufatura. Dissertação de Mestrado. Universidade Federal Fluminense, 2006.

2.5.1.2 Busca Local

A solução que é gerada na fase de construção do GRASP pode não representa uma solução ótima local para o problema, desta forma faz-se necessário realizar uma busca local a

fim de determinar uma melhor solução, dentre aquelas que se encontram na vizinhança da solução gerada inicialmente. Ou seja, o algoritmo procura encontrar soluções vizinhas de melhor qualidade que aquela gerada na etapa de construção. A busca é encerrada quando nenhuma solução melhor for encontrada na vizinhança ou quando uma solução ótima é encontrada.

Nesta etapa cada solução vizinha é avaliada através de uma medida de desempenho *f*(), verificando se alguma é melhor que a solução inicial encontrada. O Algoritmo 2.3 apresenta o pseudocódigo desta Busca Local.

```
1: Melhor_solução s* = solução_inicial
2: Obtem a vizinhança V
3: while  percorre vizinhança V  do
4:    Busca solução s' de V
5:    if  f(s') > f(s*)  then
6:       s* = s'
7:    end if
8: end while
9: return  S*
```

Algoritmo 2.3: Pseudocódigo etapa de Busca Local.

Fonte: Peres, F. S. Heurísticas GRASP para o problema de Formação de Célulasde Manufatura. Dissertação de Mestrado. Universidade Federal Fluminense, 2006.

2.5.2 ILS

A fim de obter soluções ótimas melhores e consequentemente um conjunto de regras mais adequado, a metaheurística ILS (*Iterated Local Search* – Busca Local Iterativa) será utilizada. Ela tem o intuito de melhorar a qualidade de sucessivas soluções locais. Para isto, a mesma realiza a perturbação de ótimos locais e reconsidera-os como solução inicial, caso a solução ótima gerada a partir da perturbação seja melhor que a ótima indicada inicialmente.

A ILS, proposta por Lourenço, Martin e Stützle (Lourenço et al., 2002), consiste na aplicação iterativa de um procedimento de busca local em uma solução inicial. Então, a cada iteração, é feita uma perturbação nos ótimos locais encontrados na busca. Em seguida, uma busca local é aplicada na solução perturbada gerando uma nova solução. Caso esta solução seja melhor, conforme um critério de aceitação, que a solução inicial ela passa a ser a solução atual e o processo é executado novamente, até atingir a condição de término (Algoritmo 2.4).

```
Procedimento ILS (s*, I_ILS ) //s* é a solução inicial
1. para i ← 1 até I_ILS faça
2.        s1 ← Perturbação(s*);
3.        s2 ← BuscaLocal(s1);
4.        Se f(s2) < f(s*) então
5.            s* ← s2;
6.        fim Se.
7. fim para.
8. Retorne s*;
fim ILS
```

Algoritmo 2.4: Algoritmo ILS

Fonte: Nunes, G. V. P.; Arroyo, J. E. C. Algoritmo GRASP-ILS para a minimização do atraso total do problema de programação de tarefas em uma máquina com setup time. Universidade Federal de Viçosa - MG

A metaheurística ILS é composta por três elementos básicos: Busca Local, Método de Perturbação, Critério de aceitação.

Segundo [Talbi, 2009], o método de perturbação pode ser definido como um grande movimento aleatório da solução atual. O método divide a solução corrente em duas partes, mantendo uma delas e perturbando fortemente a outra parte. Dessa forma, é possível explorar regiões que contenham melhores soluções. O tamanho da perturbação está associada a vizinhança da indivíduo em questão. Assim, conclui-se que uma pequena perturbação conduz a uma execução mais rápida da busca local.

O critério de aceitação determina se a solução gerada a partir da perturbação é aceita ou não como uma nova solução. Ou seja, define as condições de uma ótima local muito satisfatória para reproduzir a solução atual. Este critério pode ser usado para verificar o balanceamento entre a intensificação e a diversificação da busca.

2.6 CONCLUSÃO

Neste capítulo foram apresentadas as características de banco de dados geográficos, tais como tipo de dados que o compõe, forma de representação destes dados e relacionamentos existentes entre eles. Como também os principais conceitos e definições

relacionadas à Descoberta de Conhecimento em Banco de dados Relacional e Geográfico, Mineração de Dados e Mineração de dados espaciais.

Além destes conceitos, também foram mostrados os conceitos de Algoritmos Evolucionários e uma de suas ramificações, a Programação Genética. Bem como, a descrição das metaheurísticas GRASP e ILS.

O próximo capítulo apresentará um breve resumo sobre os trabalhos relacionados a esta pesquisa.

CAPÍTULO 3 TRABALHOS RELACIONADOS

Devido ao desenvolvimento dos mercados industrial, econômico e tecnológico, surgiu a necessidade de realizar previsões que auxiliem no planejamento empresarial para tomadas de decisão, melhorando o desempenho de empresas, como também a diminuição dos prejuízos. Assim existem várias atividades que suprem essas necessidades, e uma delas é a Mineração de Dados, pois a partir dela, é possível extrair conhecimento não trivial dos diferentes repositórios de dados, que se multiplicam a cada dia. Além disso, é cada vez maior o volume de dados que possuem não apenas atributos convencionais (números e textos), mas também atributos georreferenciados (aqueles que possuem coordenadas espaciais, e podem ser das formas: ponto, linha e polígono).

Atualmente, existem diversas ferramentas que dão suporte a esta atividade de extração do conhecimento a partir de bancos de dados, a saber: árvores de decisão, redes neurais artificiais e ferramentas evolucionárias (Algoritmos Genéticos, Programação Genética, dentre outros). Estas ferramentas evolucionárias são simples, robustas e bastante eficientes e por isso vem ganhando destaque na mineração de dados. Porém, existem poucas ferramentas que são capazes de extrair conhecimento a partir de dados georreferenciados, principalmente quando a base é composta pelos dois tipos de dados, convencionais e geográficos. Esta atividade de extração de conhecimento em banco de dados geográficos é chamada de mineração de dados espaciais.

Esta área de conhecimento vem sendo bastante explorada nos dias atuais, pois existem poucas ferramentas que tem a capacidade de explorar e analisar dados convencionais e geográficos. Através de pesquisas bibliográficas é possível encontrar ferramentas que fazem um pré-processamento dos dados geográficos, transformando-os em convencionais, para que sejam aplicados algoritmos de mineração de dados convencionais. A seguir, mostraremos alguns trabalhos de mineração de dados geográficos encontrados na literatura.

3.1 EXTRAÇÃO DE CONHECIMENTO EM BANCO DE DADOS GEOGRÁFICOS

[Bogorny, 2003] foi uma das autoras de uma ferramenta totalmente integrada ao Weka. A ferramenta, chamada Weka GDPM, tem por objetivo automatizar o pré-

processamento dos dados geográficos, gerando uma estrutura tabular entendida pelo Weka, onde os relacionamentos geográficos são todos mapeados em descrições textuais a fim de criar um arquivo .arff, que é o arquivo compreendido pelo Weka. Logo, através desse arquivo, pode-se utilizar qualquer algoritmo de mineração que esteja implementado no Weka.

[Pivato, 2006] também propôs uma ferramenta para realizar o pré-processamento de dados georreferenciados para aplicar algoritmo de mineração de regras de associação. Para Pivato, estudar mineração de regras de associação em dados espaciais é um desafio, pois é preciso encontrar uma boa maneira de organizar as relações topológicas e os dados. O pré-processamento era feito para determinar as relações topológicas existentes entre os dados espaciais, em que era necessário organizar os dados desnormalizando-se as tabelas para descobrir as combinações que expressavam algum tipo de conhecimento, e assim determinar os atributos mais interessantes. Depois de extrair as relações topológicas, os dados eram organizados em um determinado formato de tal forma que era possível utilizar os algoritmos de mineração de regras de associação para extrair o conhecimento desejado.

Além das ferramentas de pré-processamento, foram desenvolvidos algoritmos para a mineração de dados espaciais, para algumas atividades específicas. As mais comuns são: classificação, regras de associação e clusterização. Koperski [Koperski and Han, 1995] foi um dos pioneiros a criar algoritmos para esta atividade.

O algoritmo de regras de associação espacial proposto por Koperski [Koperski and Han, 1995] considera que os objetos espaciais podem estar associados à hierarquia de conceitos. O algoritmo é composto por cinco etapas: seleção dos objetos no banco de dados, busca por relação de proximidade, filtragem dos resultados de acordo com o suporte, especialização das relações de proximidade em relações topológicas e por fim, a obtenção das regras de associação espacial.

O algoritmo recebe como parâmetros de entrada os valores do suporte mínimo e confiança mínima e uma consulta que informa quais os objetos que serão utilizados na mineração de dados e, desses objetos, qual é o objeto principal a ser comparado com os outros, quais são os relacionamentos entre os objetos, a distância entre os objetos a serem relacionados e o objeto principal. Neste algoritmo é usado o algoritmo de mineração de regras de associação Apriori [Pivato, 2006].

Há também abordagens que utilizam as teorias de computação evolucionária, que usa conceitos de evolução das espécies, descritos por Darwin. Geralmente, estes conceitos são

usados para solucionar problemas de otimização. Na próxima seção mostraremos alguns desses trabalhos encontrados na literatura.

3.2 USO DA COMPUTAÇÃO EVOLUCIONÁRIA EM PROBLEMAS DE MINERAÇÃO DE DADOS GEOGRÁFICOS

[Dias, 2004] tenta solucionar o problema da clusterização usando a abordagem de Algoritmos Genéticos. O problema de clusterização tem por objetivo formar grupos de elementos de maneira que, os que sejam mais similares, pertençam ao mesmo subconjunto. Em seu trabalho, o autor propõe melhorias do Algoritmo Genético Tradicional, alterando seus parâmetros e a forma de inserção de novos indivíduos na população.

[Pereira, 2010] propôs um algoritmo para a atividade de classificação da mineração de dados espaciais. Conforme o autor existe poucas ferramentas que auxiliam no processo de extração do conhecimento com dados georrefenciados, e principalmente se o banco de dados for composto por dados convencionais e geográficos. O trabalho de Pereira tem por objetivo o desenvolvimento de novos algoritmos que sejam capazes de lidar com estes tipos de dados (convencionais e geográficos), a fim de extrair algum conhecimento relevante. Foi criado um algoritmo para regras de classificação chamando NGAE, baseado em Algoritmo Genético, para extrair regras a partir de dados convencionais, e outro, chamado DMGeo, baseado em Programação Genética, para extrair regras de classificação a partir de dados convencionais e geográficos.

O NGAE (*Niched Genetic Algorithm with Elitism*) foi desenvolvido para a tarefa de classificação e aplicado em um problema real de análise de gases dissolvidos (DGA) no óleo isolante de transformadores elétricos a fim de classificá-los como Normal, Falha Elétrica e Falha Térmica. O algoritmo utiliza técnicas de Nicho, Elitismo, memória cache e modela os indivíduos como cláusulas WHERE em SQL. Usa o operador de mutação, *bit by bit*. Este gera cópias, realiza a mutação e acrescenta o indivíduo novamente na população, assim, é possível preservar os indivíduos originais e aumentar a diversidade genética. O operador de cruzamento usado é o uniforme, onde todos os indivíduos possuem a mesma probabilidade de serem modificados. Este operador modifica apenas os valores numéricos dos cromossomos, além de permutar sua habilitação com probabilidade de 25%. Ao ser comparado com outros

algoritmos de clássicos de classificação (Rede Neural, SVM e Árvore de Decisão), o NGAE apresentou os melhores resultados, sendo este competitivo e robusto.

O DMGeo (*NichedGeneticProgrammingAlgorithm for Geographic Data Mining*) também usa técnicas de Nicho, Elitismo e memória cache com o intuito de melhorar o desempenho do algoritmo. Assim como o NGAE os indivíduos são modelados como cláusulas WHERE em SQL. Este algoritmo tem por objetivo tratar os dados convencionais e geográficos ao mesmo tempo, trata-se de um algoritmo inovador, pois na literatura as soluções encontradas não são capazes de lidar com os dois tipos de dados (convencionais e geográficos) simultaneamente (descrito com mais detalhes a seguir).

Além desses, há outros algoritmos para esta atividade da mineração, como regras de classificação. O SAMGA, criado por Srinivasaet al (2007), é um Algoritmo Genético Adaptativo, onde as probabilidades de mutação, cruzamento e seleção são ajustadas dinamicamente bem como o tamanho da população. Este algoritmo utilizou a abordagem Michigan. Ao ser comparado com outros algoritmos, este apresentou uma maior acurácia. Sikoraet al (2007) apresenta um framework para seleção de regras através de algoritmos genéticos. O algoritmo considera cada indivíduo como uma regra e a função *fitness* é calculada de acordo com a contagem do número padrão que casam com a regra [Pereira, 2010].

CAPITULO 4 ALGORITO DATA MINING GENETIC PROGRAMMING (DMGP)

4.1 INTRODUÇÃO

Este capítulo descreve um algoritmo baseado em Programação Genética, destinado à tarefa de classificação da Mineração de Dados Espaciais. O algoritmo, denominado DMGP (*Data Mining with Genetic Programming*), tem como objetivo principal extrair conhecimento de uma base de dados através de dados convencionais e geográficos simultaneamente. O algoritmo foi baseado no algoritmo proposto por [Pereira, 2010], chamado DMGeo, que por sua vez é um algoritmo inovador, pois, levando em consideração os que foram encontrado na literatura, não há algoritmos capazes de extrair informações a partir de dados convencionais e geográficos ao mesmo tempo. Partindo do princípio que o DMGP é uma extensão do DMGeo, um indivíduo representa um predicado lógico, definido através de uma cláusula WHERE do SQL. Para cada indivíduo é verificado o seu desempenho de acordo com uma função de *fitness*, e assim gerar populações através dos operadores de seleção e reprodução.

4.2 INDIVÍDUO

Um indivíduo é modelado de forma que seja representado um predicado lógico, definido da mesma maneira que uma cláusula WHERE da SQL. O mesmo é representado através de uma árvore de decisão, em que são encontradas as restrições e condições para classificar o padrão, conforme pode ser visualizado na Figura 4.1, que apresenta a seguinte cláusula: city.population > 200000 and crosses(rail.geom, city.geom).

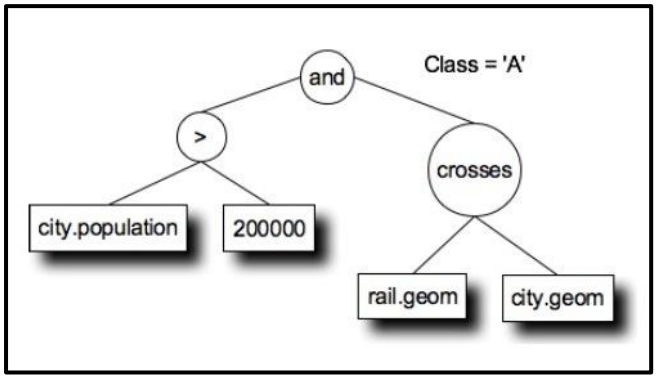

Figura 4.1: Representação de um Indivíduo do DMGeo/DMGP.

Fonte: Mineração de dados espaciais através de algoritmos evolucionários. Pereira, Marconi de Arruda, 2010.

Cada nó da árvore contém as seguintes informações:

- **Tipo:** tipo de dado que compõe o nó, que neste caso serão: booleano (lógico), numérico (real) e geográfico (ponto, linha ou polígono);
- **Corpo do nó:** restrições ou chamada de função geográfica;
- **Parâmetros:** quando o corpo do nó é uma função, o mesmo terá que receber parâmetros.

Estes nós foram classificados de duas maneiras: **Nós Funções**, que são formados por funções convencionais, tais como =, <, >, >=, <=, AND e OR, e por funções geográficas que são implementadas pelos bancos de dados com extensão geográfica. Neste trabalho foram usadas as implementações do Spatialite[1], com as funções: *contains* (contém), *covers* (cobre), *crosses* (cruza), *disjoint* (disjunto), *equals* (igual), *touches* (toca), *within* (dentro) e *distance* (distância); **Nós terminais**, que são formados por valores gerados aleatoriamente ou por atributos do banco de dados.

[1]Spatialite: Extensão espacial do banco de dados SQLite, provendo funcionalidades de banco de dados geográficos. Este banco de dados é similar ao PostGIS, Oracle Spatial e SQL Server, porém o SQLite/Spatialite não são baseados na arquitetura Cliente-Servidor. Eles adotam a arquitetura pessoal mais simples, ou seja, a engenharia SQL é diretamente embarcada na aplicação.

4.3 POPULAÇÃO INICIAL

Para gerar uma população, o usuário necessita informar ao algoritmo uma tabela alvo, a qual contém os padrões a serem classificados bem como os atributos que serão usados como parâmetro. É preciso conter, no conjunto de treinamento, os atributos que indiquem a classe à qual o padrão pertence. A geração do indivíduo pode ser visualizada no Algoritmo 4.1. A população inicial foi gerada de duas maneiras, metade dela foi seguindo o algoritmo DMGeo, ou seja, os elementos dos indivíduos foram gerados aleatoriamente e a outra metade da população foi gerada a partir da meta-heurística GRASP, descrito na seção a seguir.

Entrada: tabela alvo (T) (tabela com todos os atributos a serem classificados)

Saída: Indivíduo I

1. Crie uma lista de possíveis nós terminais, folhas_banco, com todos os atributos da tabela alvo;
2. Crie uma lista com nós não terminais, nós_internos, contendo todos os operadores (geográficos e convencionais);
3. Crie um indivíduo (árvore) aleatoriamente;
4. Gere uma classificação para o indivíduo criado;
5. Crie uma lista LCR para armazenar os operadores que são candidatos a solução;
6. Ordene a lista de acordo com o valor de classificação;
7. Enquanto cont < numMaxNosInternos faça
8. Selecione sequencialmente os elementos da lista de nós internos;
9. Se o elemento da lista **NÃO** for igual ao elemento do indivíduo então
10. Troque os elementos;
11. Gere uma nova classificação para o indivíduo com o novo elemento;
12. Armazene o novo indivíduo na LCR;
13. Fim Se
14. Selecione aleatoriamente um elemento da árvore dentre os alfa percentual melhores árvores candidatos;
15. Adicione o elemento no indivíduo que será a solução final, solução_atual;
16. Fim Enquanto
17. Enquanto não tiver solução melhor faça
18. Busque novas soluções, nova_solucao;
19. Se nova_solução for melhor que a atual
20. Solução_atual recebe nova_solução;
21. Fim Se
22. Fim Enquanto
23. Retorne I;

Algoritmo 4.1: Pseudocódigo da geração do indivíduo do DMGP.

Fonte: Próprio autor.

Para a geração da população inicial o algoritmo utiliza a meta-heurística GRASP (*Greedy Randomized Adaptative Search Procedure*) como pode ser visualizado no algoritmo 4.1.

4.3.1 GRASP

3.1.1 Construção

O algoritmo em questão inicialmente cria uma árvore completamente aleatória, e após essa construção os elementos foram trocados um a um, de acordo com a lista de nós internos, e fixados na árvore de acordo com sua função de avaliação. O indivíduo que obteve melhor classificação foi determinado como a solução final (solução inicial). Tomando como exemplo uma das bases usadas para obter os resultados para avaliação deste algoritmo, a base Íris, pode-se inferir que na primeira etapa da meta-heurística GRASP, construção gulosa, inicialmente foi determinada a lista dos nós internos e externos. Após esta determinação, foi criado um indivíduo totalmente aleatório. Sequencialmente foram testados todos os elementos, aquele que obteve a maior classificação de instâncias, era fixado na árvore, como pode ser visualizado no fluxo (Figura 4.2).

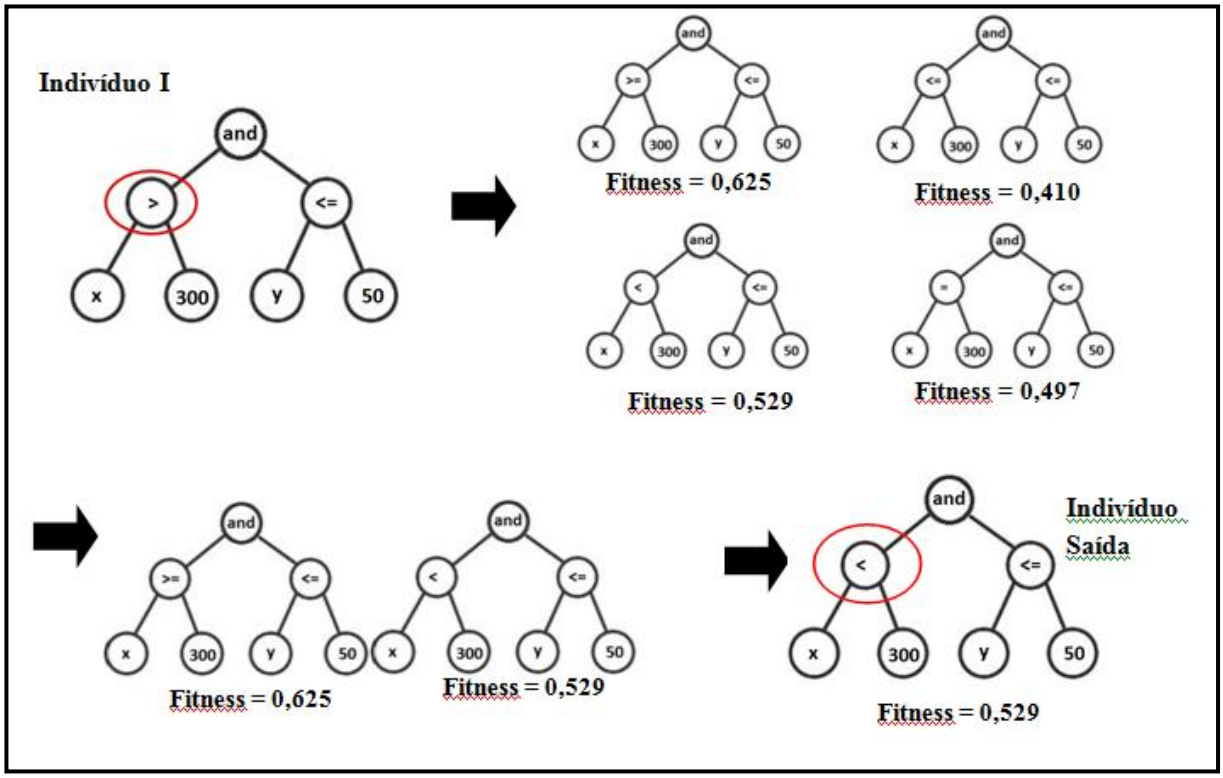

Figura 4.2: Fluxo geração de um indivíduo da população inicial, através do GRASP.

Fonte: Próprio autor.

4.3.1.2 Busca Local

A solução que é gerada na fase de construção pode não ser uma solução ótima para o problema. Desta forma faz-se necessário realizar uma busca local a fim de determinar a melhor solução ótima, dentre aquelas que se encontram na vizinhança da solução gerada inicialmente, ou seja, o algoritmo procura encontrar soluções vizinhas de melhor qualidade que aquela gerada na etapa de construção. A busca é encerrada quando nenhuma solução melhor for encontrada na vizinhança.

Nesta etapa cada solução vizinha é avaliada através de uma medida de desempenho *f*(), verificando se alguma é melhor que a solução inicial encontrada.

4.4 FUNÇÃO DE AVALIAÇÃO

Cada indivíduo é avaliado através da função *fitness*, que neste trabalho especificamente é baseada em consultas SQL, e pode ser dividida em duas etapas: (1) **Determinação dos coeficientes da Matriz de Confusão**, onde são calculados números de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos; (2) **Cálculo da performance através da função *fitness***, que usa os valores de Acurácia, Sensibilidade e Especificidade.

4.4.1 Determinação dos coeficientes da Matriz de Confusão

Para a determinação do Coeficiente de Matriz de Confusão são calculados os números de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. O número de **Verdadeiros Positivos (VP)** é o número de tuplas selecionadas nas quais as respectivas classes coincidem com a classe predita pelo indivíduo. Já os **Verdadeiros Negativos (VN)** calcula-se contando o número de tuplas que não são cobertas pelo predicado de indivíduo e não pertencem a classe do indivíduo.

Os **Falsos Positivos (FP)** são contabilizados através da identificação das tuplas obtidas pelo indivíduo que não pertencem a classe predita. E os **Falsos Negativos (FN)** são determinados através da contabilização do número de tuplas que não são obtidas pelo indivíduo, porém pertencem a classe predita pelo mesmo.

Tomando como exemplo uma base de dados, chamada Íris, que contém a classificação de um tipo de planta, uma possível classificação pode ser visualizada na Tabela 4.1. A base é classificada em três classes: setosa, versicolor e virginica e é composta por 150 instâncias, distribuída em 50 instâncias para cada classe.

Real \ Classificação	**Setosa**	**Versicolor**	**Virginica**
Setosa	48	1	0
Versicolor	0	47	3
Virginica	0	2	48

Tabela 4.1: Exemplo de uma Matriz de Confusão.

Conforme a tabela, a diagonal principal apresenta a classificação correta das instâncias: 48 da classe setosa, 47 da classe versicolor e 48 da classe virginica. Os outros valores mostram as classificações incorretas. Observe que três (3) instâncias foram classificadas como virginica, quando na verdade estas deveriam ser do tipo versicolor.

4.4.2 Cálculo da performance através da função *fitness*

Para o cálculo da função de *fitness* foram utilizados os valores de Acurácia, Sensibilidade e Especificidade, que são determinados através das seguintes fórmulas:

$$Acurácia(X) = \frac{VP+VN}{VP+FN+FP+VN} \qquad (1)$$

$$Sensibilidade(X) = \frac{VP}{VP+FN} \qquad (2)$$

$$Especificidade(X) = \frac{VN}{VN+FP} \qquad (3)$$

Já para o cálculo propriamente dito da função de avaliação a equação descrita abaixo foi utilizada:

$$f(I,X) = Acurácia(X) * Sensibilidade(X) * Especificidade(X) \qquad (4)$$

Onde *f(I,X)* é o valor de *fitness* e *I* é indivíduo que identifica padrões pertencentes à classe *X*.

Tomando como exemplo a matriz de confusão da Tabela 4.1 os cálculos dessas métricas de qualidade referentes à classe 'Virginica' são:

VP = 48

VN = 47 + 2 + 1 + 48 = 98

FP = 3 + 0 = 3

FN = 0 + 2 = 2

Com estes parâmetros, os valores de acurácia, sensibilidade e especificidade para a classe em questão são, podendo ser estendido para as demais classes:

$$Acurácia(Virginica) = \frac{48}{48 + 2 + 3 + 98} = 0{,}97$$

$$Sensibilidade(Virginica) = \frac{48}{48 + 2} = 0{,}96$$

$$Especificidade(Virginica) = \frac{98}{3 + 98} = 0{,}97$$

4.5 CRUZAMENTO

Assim como no DMGeo, o operador de cruzamento usado neste algoritmo foi o uniforme, para que todos os cromossomos possuam a mesma probabilidade de serem modificados. Este foi baseado na estrutura básica do algoritmo de Programação Genética fortemente tipada (Figura 4.3), como a seguir:

1. Selecione dois indivíduos utilizando a técnica de Roleta;
2. Selecione dois nós, um de cada indivíduo selecionado;
3. Teste se os nós escolhidos são compatíveis;
4. Caso seja, troque os dois nós selecionados entre os indivíduos;
5. Caso não seja, selecione mais dois nós.

Após realizada esta operação, é necessário verificar se os filhos (árvores) gerados são considerados válidos, ou seja, se depois de realizada a troca dos nós, os mesmos ainda continuam compatíveis com seus respectivos nós pais. Analisando o exemplo apresentado na Figura 4.3, ao selecionar os nós, "*overlaps*" e "*crosses*", foi verificado se o Pai1 é compatível com o pai do Pai2 e vice-versa. Esta verificação sendo positiva a troca é realizada com sucesso, caso contrário, haverá novas seleções de candidatos a pais.

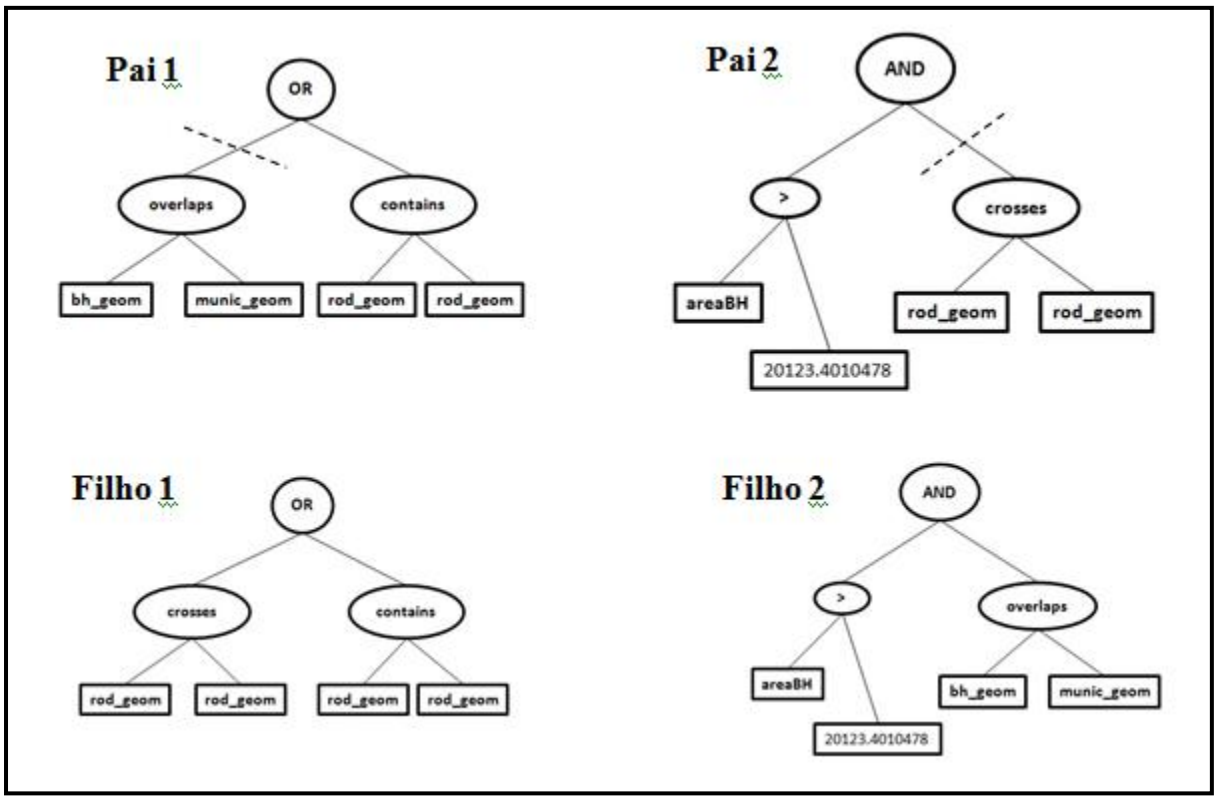

Figura 4.3: Ilustração da operação de *Crossover*.

Fonte: Próprio autor.

4.6 MUTAÇÃO

A mutação foi realizada depois do operador de cruzamento, e trata-se da troca de dois nós do mesmo tipo, como descrito a seguir.

1. Selecione um indivíduo de forma aleatória de acordo com a probabilidade determinada;
2. Selecione um nó do indivíduo escolhido;
3. Gere um nó do mesmo tipo do nó escolhido do indivíduo;
4. Troque os dois nós escolhidos, caso os tipos dos nós sejam compatíveis e que a nova árvore gerada seja válida.
5. Caso o nó escolhido do indivíduo seja um nó do tipo interno, complete os seus nós filhos com nós terminais escolhidos aleatoriamente;
6. Após modificar o indivíduo insira-o novamente na população.

Ao realizar a operação de mutação, é preciso verificar se depois de realizada a operação a árvore (o indivíduo) ainda encontra-se válida, ou seja, um nó só pode ser substituído por um nó de mesmo tipo.

Dado o indivíduo, determinado pela seguinte cláusula WHERE: **intersects (munic_geom, bh_geom) or areaBH > 20123.4010478**. Para realizar a mutação é necessário escolher aleatoriamente um dos nós da árvore para efetuar a troca. Dessa forma, o nó escolhido foi o 2 (>). Após a escolha do nó, é selecionado um nó do mesmo tipo da lista de nós internos, seleção esta feita de forma aleatória. Caso o nó da lista escolhido seja patível com o nó da árvore, ocorre a troca (Figura 4.3). Caso o nó não seja compatível, é iniciada nova seleção aleatória e assim por diante.

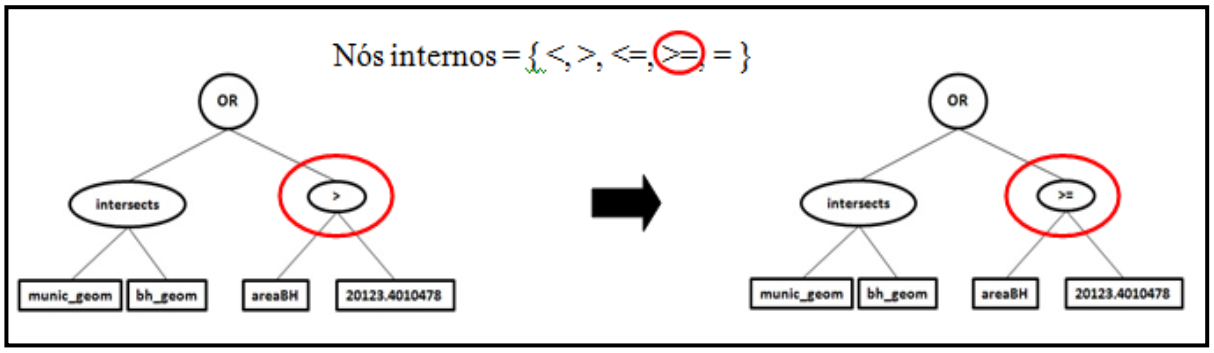

Figura 4.4: Ilustração da operação de Mutação.

Fonte: Próprio autor.

4.7 ILS

A fim de tentar obter uma solução de melhor qualidade, um mecanismo de busca local foi utilizado. Este mecanismo foi realizado através da metaheurística ILS, que tem o intuito de obter melhorias genéticas em parte da população corrente. Dessa forma, tal evolução foi obtida através do seguinte fluxo:

1. Selecionar um indivíduo (aleatoriamente) para ser melhorado de acordo com a taxa de busca local;
2. Verificar nas vizinhanças se existe um indivíduo melhor que o indivíduo escolhido;

3. Caso encontre um indivíduo melhor, substitua o indivíduo encontrado pelo da solução atual;
4. Recolocar o indivíduo na população.

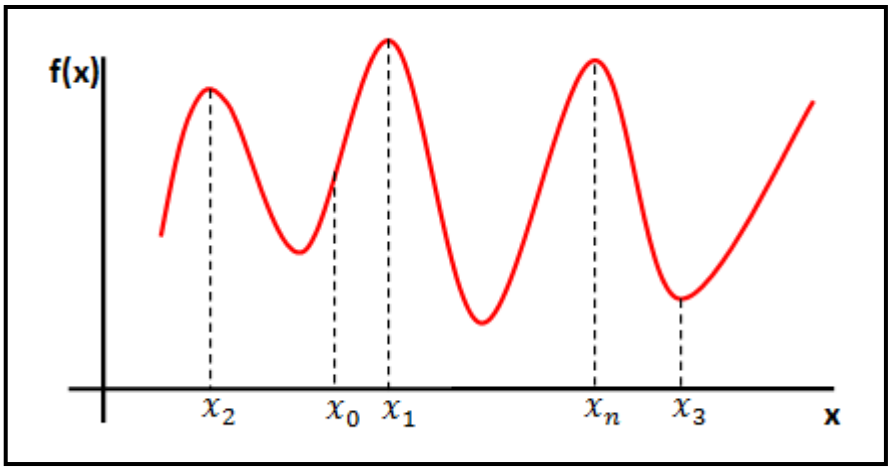

Figura 4.5: Ilustração gráfica para a execução da metaheurística ILS.

Fonte: Próprio autor.

Dada uma população P com *n* indivíduos, ilustrada pela Figura 4.2, ao selecionar um determinado conjunto de indivíduos, a metaheurística inicia a busca local. Esta percorre a vizinhança dos indivíduos escolhidos em busca de um indivíduo que seja melhor a fim de aperfeiçoar a população corrente. Observando a ilustração, percebe-se que a partir do indivíduo x_0 foram encontrados novos indivíduos melhores (também chamados de máximo local), como x_1, x_2, x_n, e piores, como x_3 .

4.8 EXPERIMENTOS E RESULTADOS

4.8.1 Problemas testes

O algoritmo proposto, DMGP, foi avaliado em dois problemas de classificação armazenados em bancos de dados disponíveis na internet: *UCI Machine Learning* e AESA. A base de dados retirada do repositório da *UCI Machine Learning* (2011), denominado Iris, é composta por dados convencionais e a base de dados, denominada Inundações, foi gerada a

partir de dados disponíveis na AESA 2011 (Agência Executiva de Gestão de Águas do Estado da Paraíba) [AESA, 2012]. Esta é composta por dados convencionais e geográficos.

A base de dados Iris é composta por 150 instâncias e classifica os tipos de planta em: setosa, versicolor e virginica. Cada uma dessas classes contém 50 instâncias. Este problema é avaliado em um conjunto de dados com quatro atributos: largura sépala, comprimento sépala, largura pétala e comprimento pétala.

O banco de dados Inundações é composto por 313 instâncias e determina se uma bacia hidrográfica possui o risco de causar enchentes, 156 instâncias da classe sim e 157 da classe não. Esta classificação foi gerada apenas a partir de estudos bibliográficos. Dentre as características usadas para gerar tal classificação tem-se quantidade de chuva ocorrida nas mediações da bacia, urbanização ou rodovias próxima, área da bacia hidrográfica.

4.8.2 Parâmetros utilizados no algoritmo

O banco de dados do UCI, Iris, foi usado a fim de obter a performance do algoritmo DMGP usando apenas dados convencionais (numéricos). Enquanto os dados da AESA utilizam os dois tipos de dados (convencionais e geográficos). Desta forma, pode-se afirmar que uma das principais contribuições do algoritmo proposto é saber explorar adequadamente as propriedades dos problemas onde os conjuntos de dados são formados por atributos geográficos e convencionais.

Além do DMGP, os conjuntos de dados foram classificados também com uma versão do algoritmo DMGeo, proposto por Marconi Pereira (2010), que não utiliza metaheurísticas.

Cada execução do algoritmo foi repetida 5 vezes, usando cada um dos bancos de dados. Os algoritmos usaram o tamanho da população = 400, número de gerações = 100, a probabilidade de *crossover* = 40%, a probabilidade de mutação = 2% e a seleção de indivíduos a serem melhorados via busca local = 10%.

4.8.3 Resultados e Análises

Esta seção apresenta uma comparação entre resultados obtidos com o DMGP e uma versão do DMGeo. A Tabela 4.2 e o Gráfico 4.1 mostram os resultados de uma forma geral

para cada algoritmo testado para os bancos de dados Iris e Inudações. O índice geral consiste numa média dos resultados obtidos em cada classe. Como mencionado anteriormente, a média dos resultados foi calculada com base nos resultados das 5 execuções realizadas para cada cenário. Então podemos inferir que usando o algoritmo DMGP foram classificadas corretamente 61% das instâncias da base de dados Irís e 66% da base de dados Inundações, enquanto o DMGeo classificou corretamente apenas 55% e 60% das bases Iris e Inundações, respectivamente.

Base	Versão DMGeo	DMGP
Iris	55%	61%
Inundações	60%	66%

Tabela 4.2: Índice Global.

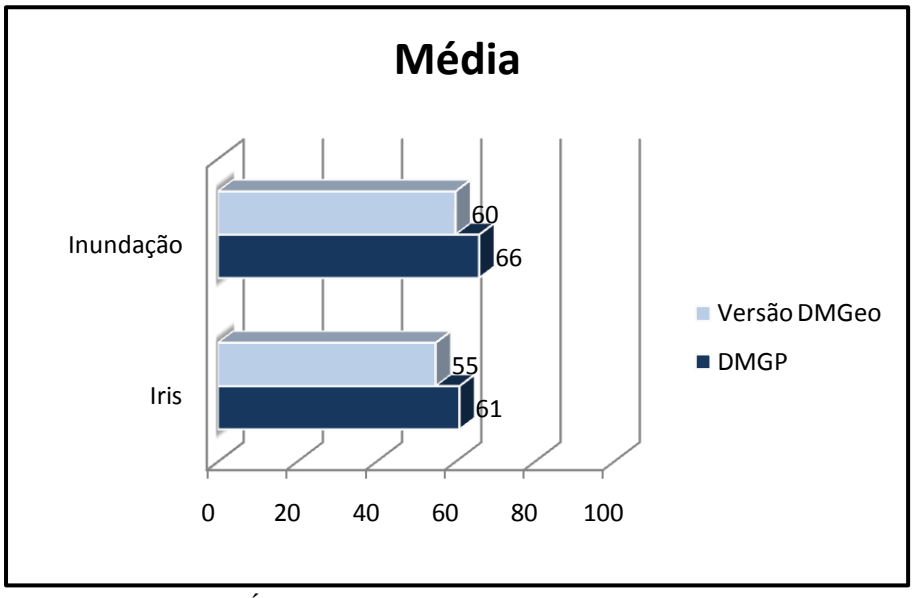

Gráfico 4.1: Média do Índice Global.

Apesar de ter obtido resultados melhores que a versão do DMGeo, o DMGP gastou um maior tempo de processamento para chegar em tal resultado. Ao utilizar a metaheurística GRASP em que uma de suas etapas é a construção gulosa, levou um tempo considerável para executar o processamento ao construir cada elemento dos indivíduos da população. Isto

ocorreu principalmente com a base de dados que possui os dois tipos de dados (convencionais e geográficos), pois os dados geográficos demandam ainda mais processamento e análise.

Além da GRASP, o mecanismo de Busca Local também demanda mais processamento por realizar a busca por toda a vizinhança de uma determinada solução a fim de encontrar uma solução que seja ainda mais viável para o problema em questão. Tecnicamente, buscando uma solução ótima melhor que a solução ótima já determinada. Com o Gráfico 4.2, é possível visualizar a variação do tempo, em minutos, gasto na execução de cada algoritmo.

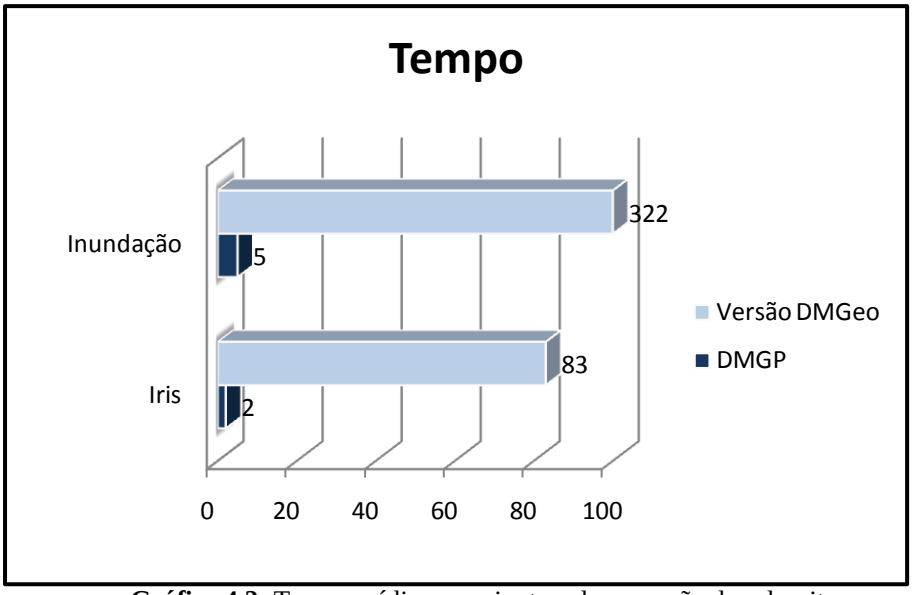

Gráfico 4.2: Tempo médio, em minutos, de execução dos algoritmos.

A Tabela 4.3 mostra a performance obtida para cada classe dos conjuntos de dados usados. De acordo com a tabela pode-se afirmar que o algoritmo DMGP apresentou melhores resultados se comparado ao DMGeo, levando em consideração a classificação para cada classe das bases utilizadas. A base de dados Iris teve 76% das instâncias classificadas corretamente para a classe "setosa" enquanto a base Inundações teve 73% das instâncias classificadas corretamente para a classe "não".

Mesmo mostrando alguns resultados insatisfatórios, de maneira geral, é possível afirmar que a análise com o DMGP obteve resultados positivos quando comparado ao DMGeo, pois o algoritmo classificou corretamente mais instâncias, provando que o uso das

metaheurísticas resulta em indivíduos melhores e, consequentemente, soluções próximas a soluções ótimas.

Base	Classes	Versão DMGeo	DMGP
Iris	Setosa	45%	76%
	Versicolor	63%	59%
	Virginica	56%	53%
Inundações	Sim	60%	59%
	Não	60%	73%

Tabela 4.3: Índice Global da classificação de cada classe.

A Tabela 4.4 apresenta os melhores e piores resultados de cada base dentre as cinco execuções realizadas. Observa-se que ocorreu uma grande variação entre o melhor e o pior resultado. É possível notar que ocorreu dentre as cinco execuções o caso perfeito em que o algoritmo classifica corretamente todas as instâncias.

	Classificação	Versão DMGeo		DMGP	
		Corretas	Incorretas	Corretas	Incorretas
Iris	Melhor	91%	9%	87%	13%
	Pior	38%	62%	32%	68%
Inundações	Melhor	90%	10%	100%	0%
	Pior	36%	64%	31%	69%

Tabela 4.4: Índice Global do melhor / pior resultado do algoritmo DMGP.

Além da validação do algoritmo através da comparação com o DMGeo, foi realizada também uma análise sobre a influência das taxas de mutação, *crossover* e da seleção dos indivíduos da busca local, na execução do algoritmo, para cada uma das bases utilizadas nos testes. Os parâmetros foram variados da seguinte forma:

- Mutação = 1, 2, 4;
- *Crossover* = 20, 40, 60;
- Busca Local = 5, 10, 15.

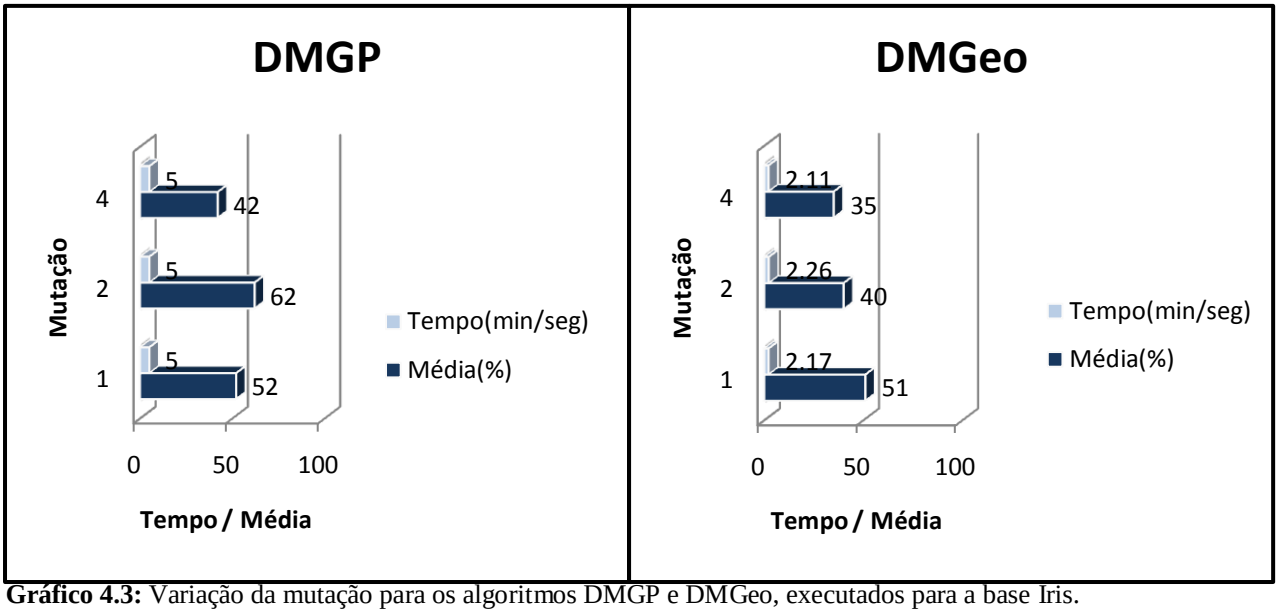

Gráfico 4.3: Variação da mutação para os algoritmos DMGP e DMGeo, executados para a base Iris.

Os gráficos 4.3 e 4.4 apresentam a variação dos resultados ao executar tais cenários. O Gráfico 4.3 mostra a variação da mutação para os dois algoritmos executados com a base Irís. Através deles é possível perceber que o algoritmo DMGP gerou uma melhor solução quando a mutação foi igual a 2, com o tempo de execução de 5 minutos. Enquanto para o algoritmo DMGeo, a melhor solução pôde ser encontrada quando a mutação foi igual a 1, obtendo um tempo de execução igual a 2 minutos. Também pode-se observar que para os dois algoritmos a mutação teve uma variação parecida, ou seja, o algoritmo se comportou de maneira semelhante ao ser executados com valores diferentes para este parâmetro.

O Gráfico 4.4 apresenta a variação da mutação para os dois algoritmos executados com a base Inundações. É possível observar que o algoritmo DMGP gerou uma melhor solução quando a mutação foi igual a 2, executando no tempo de 302 minutos. Já o algoritmo DMGeo, obteve a melhor solução quando executado com a mutação igual a 4 com um tempo de 85 minutos. Assim como na execução da base Iris, o algoritmo ao executar a base Inundações também se portou de maneira análoga, mostrando que não ocorreu uma enorme variação entre as execuções. Porém, quando é observado o tempo gasto para cada algoritmo, ocorre uma variação considerável, pois o DMGP apesar de gerar soluções melhores, possui um processamento maior por ter sido implementado com metaheurísticas, descritas anteriormente.

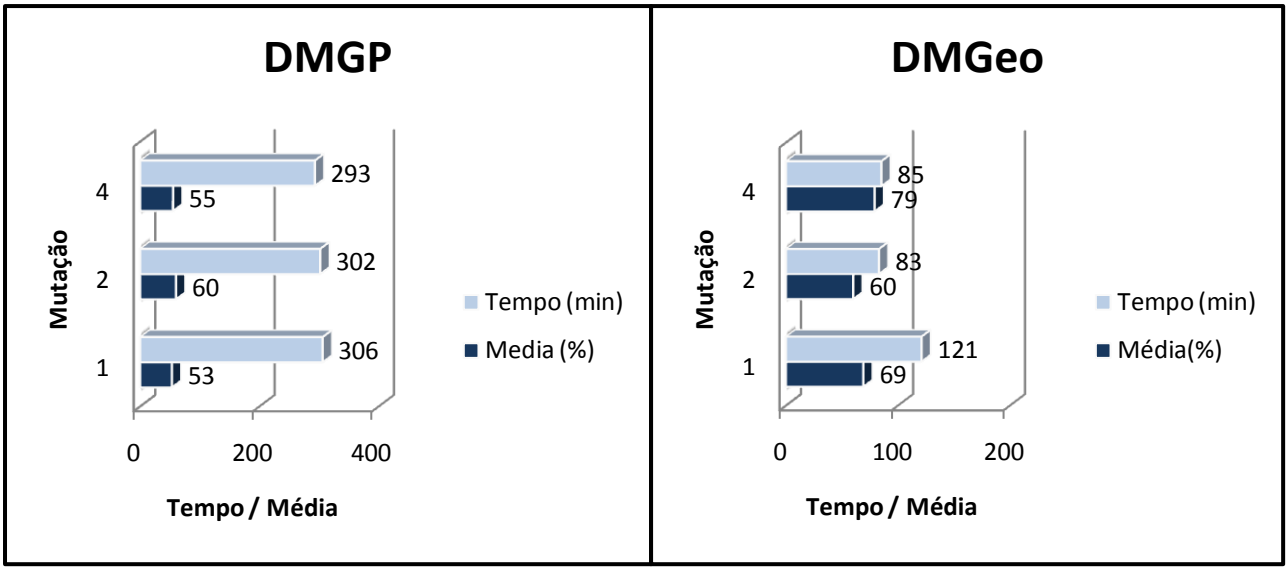

Gráfico 4.4: Variação da mutação para os algoritmos DMGeo e DMGP, executados para a base Inundações.

O Gráfico 4.5 apresenta a variação do *crossover* para os dois algoritmos executados para a base Irís. Através deles é possível perceber que o algoritmo DMGP obteve a melhor solução quando executado com o *crossover* igual a 20, com uma média de 55% de instâncias classificadas corretamente em um tempo de 4 minutos. No gráfico também é possível observar que diferente do DMGP, o algoritmo DMGeo apresentou uma solução melhor quando executado com o *crossover* foi igual a 40, classificando corretamente, em média, 55% das instâncias da base em um tempo de 2 minutos. Os algoritmos se comportoram de maneira semelhante com a variação do parâmetro *crossover*.

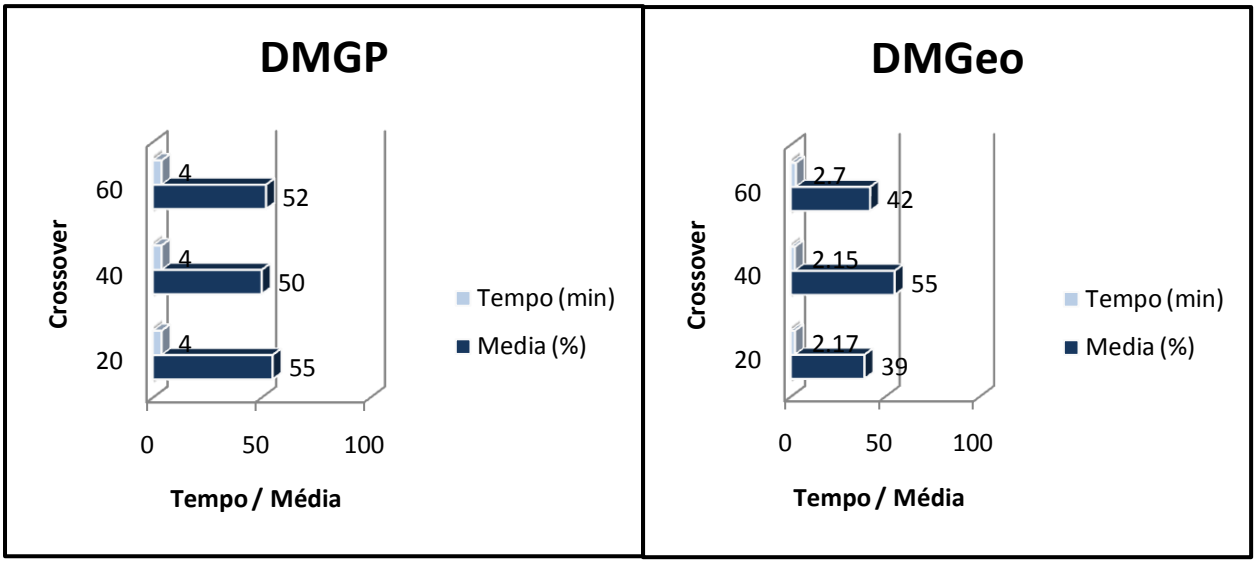

Gráfico 4.5: Variação da *crossover* para os algoritmos DMGeo e DMGP, executados para a base Iris.

O Gráfico 4.6 apresenta a variação do *crossover* para os dois algoritmos executados para a base Inundações. Nota-se que o algoritmo DMGP gerou soluções melhores quando o *crossover* foi igual a 40, classificando corretamente 63% das instâncias da base. Já o algoritmo DMGeo gerou soluções melhores quando executado com crossover igual a 20, classificando 64% das instâncias da base. Também para esta variação nota-se a significativa diferença dos tempos de execução para cada algoritmo. O DMGP apesar de ter gerado soluções melhores com o crossover = 40, obteve um tempo de execução de 305 minutos. Enquanto o DMGeo obteve seu maior tempo de execução, 124 minutos, quando o crossover = 60.

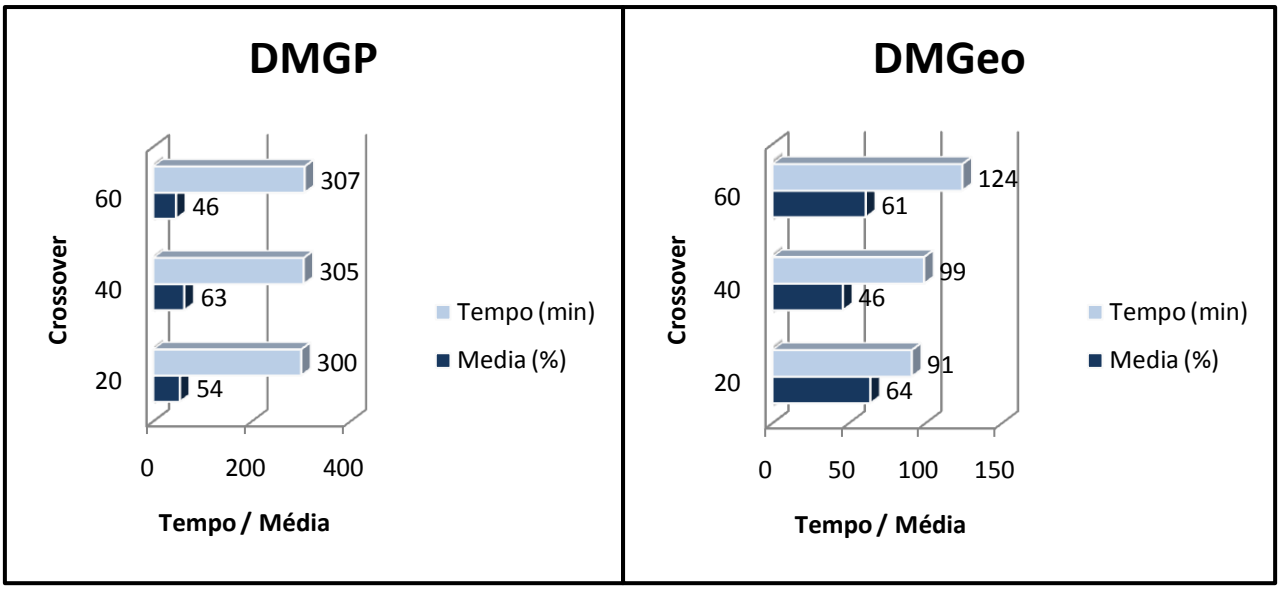

Gráfico 4.6: Variação da *crossover* para os algoritmos DMGeo e DMGP, executados para a base Inundações.

O Gráfico 4.7 mostra a variação da busca local para o algoritmo DMGP quando executado com a base de dados Íris. O algoritmo apresentou melhor solução quando executado com a busca local = 10, classificando 61% das instâncias da base, executando em 5 minutos. Também é possível observar que, ao executar o algoritmo com as três variações do parâmetro, o algoritmo não se comportou de maneira muito distinta.

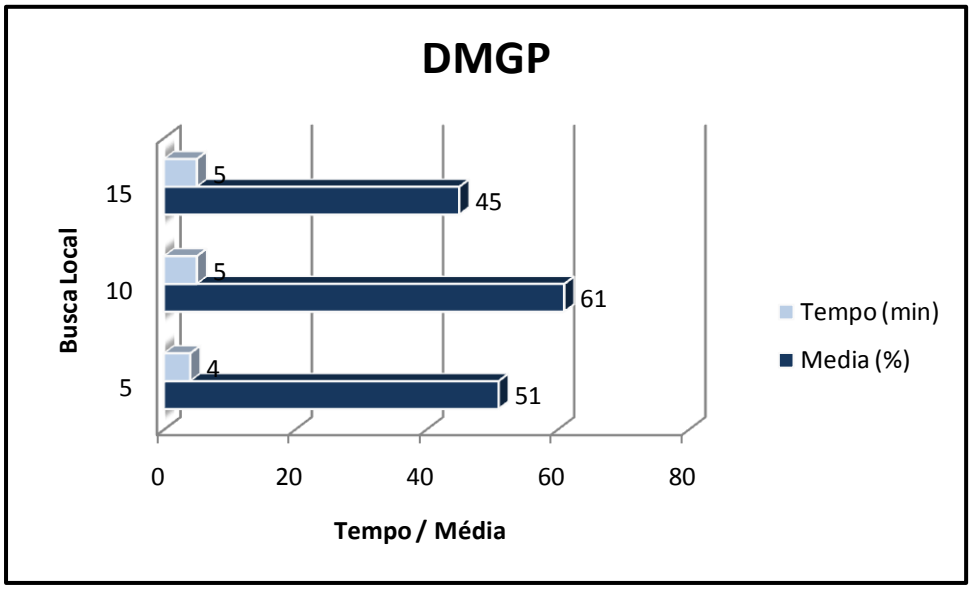

Gráfico 4.7: Variação da Busca Local para o algoritmo DMGP, executado para a base Íris.

O Gráfico 4.8 apresenta a variação da busca local para o algoritmo DMGP quando executado com a base de dados Inundações. O algoritmo classificou corretamente 66% das instâncias quando executado com a busca local = 10, apresentando sua melhor solução. Porém em um alto tempo de execução, 322 minutos. Assim como nas demais análises, para esta variação o algoritmo também se comportou de maneira semelhante, mostrando apenas uma variação maior para o tempo de execução.

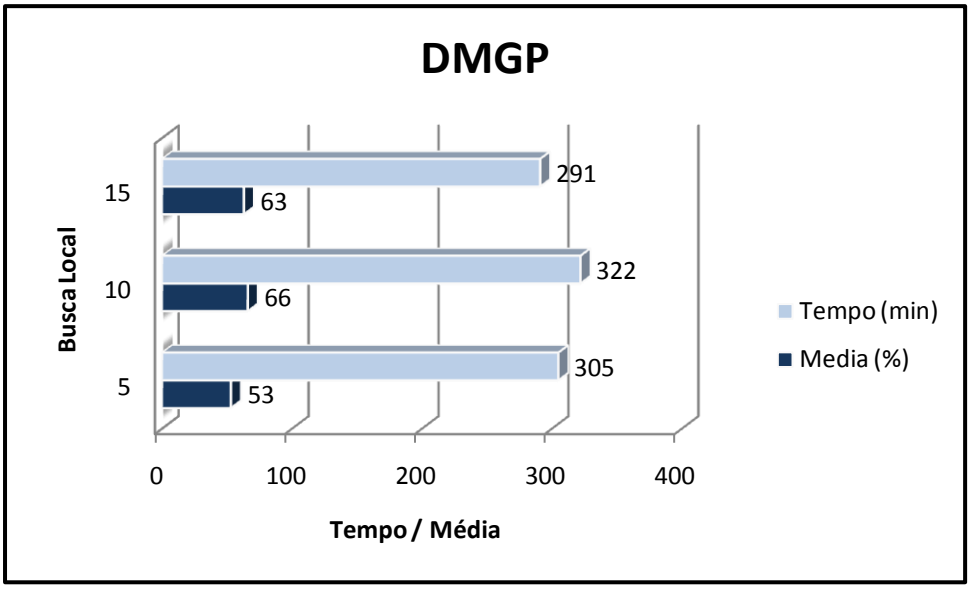

Gráfico 4.8: Variação da Busca Local para o algoritmo DMGP, executado com a base Inundações.

4.9 CONCLUSÃO

Este capítulo apresentou um algoritmo evolucionário aplicado em problemas de classificação com dados geográficos. O algoritmo usa metaheurísitcas a fim de prover melhoria genética a um subconjunto de indivíduos da população corrente. Os indivíduos foram modelados de acordo com uma cláusula WHERE da linguagem SQL. Para avaliar o desempenho do algoritmo foram usados problemas de classificação. A partir da análise realizada é possível concluir que o algoritmo proposto obteve melhores resultados se comparado ao DMGeo. Mostrando que o uso de metaheurísticas constrói indivíduos melhores, consequentemente apresentando regras de classificação mais interessantes e precisas.

CAPÍTULO 5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Nesta dissertação foi realizado um estudo de problemas de mineração de dados espaciais. A descoberta de conhecimento em bancos de dados é bastante ampla e relevante porque é a partir dela que é possível inferir informações sobre determinadas bases de dados.

O algoritmo DMGP foi proposto como uma extensão do DMGeo, onde etapas foram modificadas. Na geração da população inicial, temos parte da mesma construída pela aplicação da metaheurística GRASP, e na fase de evolução das populações foi proposta a inserção de um mecanismo de Busca Local usando ILS, que promove a melhora genética da parte da população corrente.

Foram utilizadas duas bases de dados para realizar os experimentos a fim de validar o algoritmo proposto. Uma base de dados composta apenas por dados convencionais, chamada Íris, e outra base contendo dados convencionais e geográficos, denominada Inundações. A base Íris foi retirada do Repositório *UCI Machine Learning Repository* e a base Inundações foi construída a partir de estudos bibliográficos e bases de dados retiradas do site da AESA.

Apesar de o algoritmo DMGP apresentar um tempo de execução maior que o DMGeo, o que justifica-se pelo fato de obter mais processamento por causa das metaheurísticas utilizadas, foi observado que o algoritmo apresentou melhores resultados, através de analise realizada, provando assim, ser mais eficiente e robusto. Ao comparar com o algoritmo DMGeo, os resultados computacionais demonstraram ganhos significativos de desempenho para o algoritmo DMGP, quando avaliados sobre os bancos de testes: Iris e Inundações. Além disso, também foi possível observar o comportamento do algoritmo ao realizar variações nos parâmetros de entrada: mutação, *crossover* e busca local. O algoritmo se comportou de maneira semelhante a cada variação realizada, como pode ser visualizado na seção 4.8.3.

Como descrito anteriormente o uso das metaheurísticas GRASP e ILS acarretaram em soluções ótimas melhores nos experimentos realizados, pois através da GRASP foi possível determinar indivíduos melhores para compor a população inicial e o uso da ILS proporcionou encontrar soluções melhores que as determinadas a priori, através de buscas realizadas nas vizinhanças das soluções.

Como trabalhos futuros, realizar uma análise mais completa através da execução de testes com outros algoritmos como, por exemplo, J48, SVN, dentre outros, buscando efetuar novas comparações e, consequentemente, obter uma validação mais precisa do algoritmo.

Além das novas comparações, também é necessário analisar o desenvolvimento do algoritmo a fim de diminuir o processamento do mesmo, consequentemente, o tempo de execução, sem interferir em seu desempenho.

REFERÊNCIAS

[AESA] AESA – Agência Executiva de Gestão das Águas do Estado da Paraíba. Disponível em http://www.aesa.pb.gov.br/ (acessado em Novembro de 2011)

[Bogorny, 2003]Borgony,V. ; **Algoritmos e Ferramentas de descoberta de conhecimento em banco de dados geográficos.**Universidade Federal do Rio Grande do Sul. Porto Alegre, 2003.

[Bogorny, 2006] Bogorny, V. ***"Enchancing spatial association rules mining in geographic databases."*** Tese de Doutorado. Porto Alegre: Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande do Sul, 2006.

[Câmara, 2005]Câmara, G.; **Banco de Dados Geográficos.** Editora MundoGEO, Curitiba, 2005. Disponível em: http://www.dpi.inpe.br/gilberto/livros.html.

[Câmara, Freitas e Neves, 2001] Câmara, G.; Freitas, C. C.;Neves, M. C. **Mineração de Dados em grande Banco de Dados Geográficos**, Relatório Técnico. INPE, 2001.

[Castanheira, 2008] Castanheira, L. G.; **Aplicação de Técnicas de Mineração de Dados em Problemas de Classificação de Padrões**. Dissertação de Mestrado. Universidade Federal de Minas Gerais. 2008.

[Cavalcanti, 2005] Cavalcanti, V. M. B.; **Um ambiente Visual de Consulta a Banco de Dados Espaço-Temporal**. Dissertação de Mestrado. Universidade Federal de Campina Grande, 2005.

[Côrtes, Porcaro, Lifschitz, 2002] Lifschitz, S.; Côrtes, S.; Porcaro, R.;**Mineração de dados, Funcionalidades, Técnicas e Abordagens**. ISSN 0103-9741, PUC-Rio 2002.

[Darwin, 2003] Darwin, C.; **A Origem das Espécies**.

[Dias, 2004] Dias, C. R. **Algoritmos Evolutivos para o Problema de Clusterização de Grafos Orientados: Desenvolvimento e Análise Experimental**. Dissertação de Mestrado. Universidade Federal Fluminense, Niterói - RJ. 2004.

[Egenhofer e Herring, 1995] Egenhofer, M.; Herring, J.; ***Categorizing Binary Topological Relations***.*International Journal of Geographical Information Systems*, 1995.

[Elmasri e Navathe, 2005] Elmasri, R. e Navathe, S. B. **Sistemas de Banco de Dados**.Editora Pearson, São Paulo, BRA. 2005.

[Goldberg, 1989] Goldberg, D. E. ***Genetic Algorithms in Search, Optimization & Machine Learning***.Ed. Addison Wesley.Canadá, 1989 (2006).

[Han e Kamber, 2006]Han, J.; Kamber, M.; ***Data Mining Concepts and Techniques***.*Second Edition.San Francisco: Morgan Kaufmann Publishers*. 2006.

[Koperski and Han, 1995] Koperski, K.; Han J.;***Discovery of Spatial Association Rules inGeographic Information Database.****School of Computing Science. Simon Frases University*.1995.

[Koza, 1992] Koza, John R. ***Genetic Programming: On the Programming of Computers by Means of Natural Selection.***Editora Bradford Book, Londres, 1992.

[Lourenço et al, 2002] Lourenço H.R. , Martin, O. and Stützle T. ***Iterated local search***. *In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, volume 57 of International Series in Operations Research & Management Science, pages 321-353. KluwerAcademicPublishers, Norwell, MA.* 2002.

[Mello, Silva, Souza 2007] Mello, C. ; Silva, G. Z. ; Souza, J. M.;**Extensão do WEKA para Métodos de Agrupamento com Restrição de Contigüidade.**Geoinfo - XI Simpósio Brasileiro de Geoinformática, Campos do Jordão, 2007.

[Pedrosa, 2010] Pedrosa, K. A. **GeoMiningVisualQL: Uma linguagem de consulta visual para Mineração de Dados Geográficos**. Dissertação de Mestrado, João Pessoa - PB. 2010.

[Peres, 2006] Peres, F. S. H**eurísticas GRASP para o problema de Formação de Células de Manufatura.** Dissertação de Mestrado. Universidade Federal Fluminense, 2006.

[Pereira, 2010] Pereira, M. A.; **Mineração de dados espaciais através de Algoritmos Evolucionários**. Exame de Qualificação. Universidade Federal de Minas Gerais, 2010.

[Pivato, 2006] Marina A. P.; **Mineração de regras de associação em dados georreferenciados.** Dissertação de Mestrado. Instituto de Ciências Matemáticas e de Computação – ICMC – USP.2006.

[Rodrigues, 2002] Rodrigues, E. L. M.;**Evolução de Funções em Programação Genética orientada a gramáticas.**Dissertação de Mestrado. Universidade Federal do Paraná, Curitiba, 2002.

[Samuel, 1959] Samuel, A. L. ***Some studies in machine learning using the game ofcheckers***. *IBM JournalofResearchandDevelopment*. 1959.

[Santos, 2008] Santos, J. S. **Mineração de dados utilizando algoritmos genéticos.** UniversidadeFederal da Bahia. Salvador, Bahia, 2008.

[Soares, 2008] Soares, G. L. **Algoritmos Determinístico e Evolucionário Intervalares para Otimização Robusta Multi-Objetivo.** Tese de Doutorado.Universidade Federal de Minas Gerais, 2008.

[Souza, 2006] Souza, L. V.; **Programação Genética e Combinação de Preditores para previsão de Séries Temporais.** Tese de doutorado. Universidade Federal do Paraná, Curitiba, 2006.

[Talbi, 2009] Talbi, E.; ***Metaheuristics from design to implementation***.Publicado por *John Wiley& Sons, Inc. Hoboken, New Jersey*, 2009.

[Trindade, 2005] Trindade, V. A. **Desenvolvimento e Análise Experimental da Metaheuristica GRASP para um problema de planejamento de Sondas de Manutenção.** Dissertação de Mestrado. Universidade Federal Fluminense - Niterói - RJ, 2005.

[UCI] **UCI Machine Learning Repository**. University of California, School of Information and Computer Science, Irvine, CA. Disponível em: http://archive.ics.uci.edu/ml/ (acessado em Fev 2012).