AMÍLCAR SOARES JÚNIOR

X-GAT: UMA FERRAMENTA BASEADA EM XML PARA OTIMIZAÇÃO COM ALGORITMOS GENÉTICOS

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

João Pessoa

2011

AMÍLCAR SOARES JÚNIOR

X-GAT: UMA FERRAMENTA BASEADA EM XML PARA OTIMIZAÇÃO COM ALGORITMOS GENÉTICOS

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

Área de concentração: Sistemas de Computação
Orientador: Prof. Dr. Gustavo Henrique Matos Bezerra Motta.

João Pessoa

2011

S676x Soares Júnior, Amílcar.

1.1.1.1 X-GAT : uma ferramenta baseada em XML para otimização com algoritmos genéticos / Amilcar Soares Júnior. - - João Pessoa: [s.n.], 2011.

129f. : Il.

Orientador: Gustavo Henrique Matos Bezerra Motta.
Dissertação (Mestrado) – UFPB/CCEN.

1. Informática. 2. Engenharia de software. 3. Ferramentas de Otimização. 4. Inteligência artificial.

UFPB/BC *CDU: 004(043)*

Ata da Sessão Pública de Defesa de Dissertação de Mestrado do AMILCAR SOARES JUNIOR, candidato ao Título de Mestre em Informática na Área de Sistemas de Computação, realizada em 04 de março de 2011.

Aos quatro dias do mês de março do ano dois mil e onze, às dez horas, na Sala de Reuniões do Centro de Ciências Exatas e da Natureza da Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Computação Distribuída*", o Sr. Amilcar Soares Júnior. A comissão examinadora composta pelos professores doutores: Gustavo Henrique Matos Bezerra Motta (DI - UFPB), Orientador e Presidente da Banca Examinadora, Lucídio dos Anjos Formiga Cabral (DI-UFPB), e Natasha Correia Queiroz Lino (DI-UFPB) como examinadores internos e Celso Augusto Guimarães Santos (UFPB) e Valéria Cesário Times (UFPE) como examinadores externos. Dando início aos trabalhos, o Prof. Gustavo Henrique Matos Bezerra Motta, cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado "*X-GAT: UMA FERRAMENTA BASEADA EM XML PARA OTIMIZAÇÃO COM ALGORITMOS GENÉTICOS*". Concluída a exposição, o candidato foi argüido pela Banca Examinadora que emitiu o seguinte parecer: "*aprovado*". Assim sendo, deve a Universidade Federal da Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para constar, o professor Lucídio dos Anjos Formiga Cabral, Sr. Vice-Coordenador do PPGI, lavrou a presente ata, que vai assinada por ele, e pelos membros da Banca Examinadora. João Pessoa, 04 de março de 2011.

Lucídio dos Anjos Formiga Cabral

Prof. Dr. Gustavo Henrique M. Bezerra Motta
Orientador (DI-UFPB)

Prof. Dr. Lucídio dos Anjos Formiga Cabral
Examinador Interno (DI-UFPB)

Profª. Dra. Natasha Correia Queiroz Lino
Examinador Interno (DI-UFPB)

Prof. Dr. Celso Augusto Guimarães Santos
Examinador Externo (UFPB)

Profª Dra.Valéria Cesário Times
Examinador Externo (UFPE)

Aos meus pais e minhas irmãs. Seu amor e apoio me deram forças para mais uma vitória.

Agradecimentos

Primeiramente, gostaria de agradecer a Deus por todas as bênçãos derramadas sobre a minha família nesses últimos anos e por Ele sempre se mostrar presente nas pequenas coisas da vida. Em seguida, aos meus pais, Amilcar e Teresa, por todo o amor, carinho, proteção, estímulo, educação e dedicação em todos os anos da minha vida. Gostaria também de agradecer as minhas irmãs, Thaíse e Thaiane, pela tolerância, carinho e compreensão nesses últimos anos. À minha namorada, Cynthia M. L. Freire por todo carinho, amor e paciência durante esta etapa da minha vida. Agradeço especialmente ao professor e tio Hamilton Soares, por todos os conselhos e paciência desde a minha opção pelo curso de Ciências da Computação.

Agradeço também aos meus amigos Francisco Melo (Kiko), Hugo Duque, Felipe Crisanto, Thiago Fernandes (coloral) e Yuri Morais por todos os momentos de descontração, discussões e apoio, mostrando realmente o verdadeiro significado da palavra amizade.

Por último, agradeço a todos os professores do Departamento de Informática pela dedicação e estímulo empreendidos ao longo desses quatro anos. Ao professor Gustavo Motta pela primeira oportunidade como monitor de disciplina e por toda atenção e orientação até hoje. Agradeço especialmente aos meus orientadores e amigos, Cristiano das Neves Almeida e Celso Augusto Guimarães, e aos companheiros do laboratório LARHENA por acreditarem no meu trabalho e contribuírem muito para a minha formação profissional.

Sumário

Lista de Figuras e Gráficos

Lista de Tabelas

Lista de Quadros

Lista de Acrônimos

AG	Algoritmo Genético
AGs	Algoritmos Genéticos
PG	Programação Genética
EEs	Estratégias Evolucionárias
PE	Programação Evolucionária
XML	eXtensible Markup Language
API	Application Programming Interface
X-GAT	XML based Genetic Algorithm Toolkit
JEP	Java Expression Parser
SCE-UA	Shuffled Complex Evolution
WESP	Watershed Erosion Simulation Program
SUDENE	Superintendência do Desenvolvimento do Nordeste
ORSTOM	French Office of Scientific Research and Technology for Overseas Development
UFPB	Universidade Federal da Paraíba
CB-SMoT	Clustering-Based Stops and Moves of Trajectories
DB-SMoT	Direction-Based Stops and Moves of Trajectories
ROC	Receiver Operating Characteristic
ROOCH	Receiver Operating Characteristic Convex Hull
LGPL	GNU Library or Lesser General Public License

Resumo

SOARES JÚNIOR, A. **X-GAT: Uma Ferramenta Baseada em XML para Otimização com Algoritmos Genéticos**. 2011. (129 p.) Dissertação (Mestrado) – Programa de Pós-Graduação em Informática, Departamento de Informática, Universidade Federal da Paraíba, João Pessoa, Brasil.

Em vários campos da ciência nos deparamos com problemas de otimização. Muitas abordagens foram propostas para resolver tais problemas, incluindo o uso de algoritmos evolucionários (AEs) e algoritmos genéticos (AGs). Entretanto poucos trabalhos tentaram criar ferramentas genéricas, capazes de serem reutilizadas em vários problemas distintos. A maior parte dos trabalhos vistos na literatura mostram diversas implementações de AGs ou AEs, sendo estas dirigidas a um propósito específico. Verifica-se assim a necessidade da criação de ferramentas de otimização que sejam de propósito geral. O objetivo deste trabalho é desenvolver uma ferramenta de otimização, chamada X-GAT (**X**ML based **G**enetic **A**lgorithm **T**oolkit), que resolva problemas de otimização através do uso AGs, sendo esta ferramenta de propósito geral, configurável, portável, expansível, que possa funcionar em sistemas heterogêneos e que também seja um framework de suporte a otimizações com AEs. Diante disso, o esforço para realizar uma otimização se concentra na forma de configurar a ferramenta, e não em como a técnica deve ser implementada. A partir de algumas técnicas computacionais utilizadas algumas partes do algoritmo podem ser abstraídas, evitando que se gaste muito tempo em codificação e validação da técnica. Para que estes objetivos fossem alcançados, foram utilizados no seu desenvolvimento: a linguagem de programação Java e sua API de reflexão, a linguagem de descrição de dados eXtensible Markup Language (XML) e padrões de projeto (*design patterns*) de software. A fim de validar e verificar que a ferramenta atendeu a todos os princípios propostos, foram selecionados problemas de otimização de áreas distintas. Primeiramente, são mostradas otimizações de algumas funções matemáticas a fim de validar o algoritmo. Posteriormente a ferramenta é aplicada a problemas de otimização de parâmetros de entrada de modelos hidrológicos de transformação de chuva em vazão. Por fim, a ferramenta é aplicada no processo de otimização de parâmetros de entrada de algoritmos de clusterização de pontos de trajetórias de objetos móveis, que são usados com freqüência para adição de informações semânticas em dados espaciais. A partir dos resultados obtidos em diferentes áreas da ciência, pode-se concluir que a ferramenta é bastante flexível e robusta, podendo ser aplicada de forma simples e prática, desde que configurada de forma adequada.

Palavras-chave: engenharia de software, inteligência artificial bio-inspirada, otimização.

Abstract

SOARES JUNIOR, A. **X-GAT: An XML-based Genetic Algorithm Toolkit for Optimization Problems**. 2011. (129 p.) Dissertation (Masters) – Programa de Pós-Graduação em Informática, Departamento de Informática, Universidade Federal da Paraíba, João Pessoa, Brazil.

In many different fields of science we are faced to optimization problems. Many approaches have been proposed to solve such types of problems, including the use of Evolutionary Algorithms (AEs) and Genetic Algorithms (GAs). However, just a low quantity of works tried to create generic tools, capable to be reused in many different problems. Most of the current proposals show different implementations of GAs or AEs applied to specific purposes. Raising all these assertions, it is verified the need for the development of optimization tools that are capable of solving any kind of optimization problems. In such context, the main objective of this work is the development of an optimization tool, named X-GAT (XML based Genetic Algorithm Toolkit), that is: capable of solving any kind of optimization problem; configurable; operating system portable; extensible; a framework that helps the implementation of AEs; and can be used to build heterogenic systems. Using some computational techniques, parts of the algorithm can be abstracted, preventing that much time is spent on coding and validating the optimization technique. Aiming to achieve such objectives, some tools and techniques were used: Java programming language and its reflection API; the data description language XML (eXtensible Markup Language); and software design patterns. In order to verify and validate that the developed tool attended the proposed objectives, many different optimization problems were chosen. First, it is shown the optimization of some mathematical functions that have known optimum value. Then, the X-GAT tool is applied to the calibration of for rainfall-runoff models input parameters, which are common problems in hydrology. Finally, the X-GAT tool is used in an optimization process of input parameters of clustering algorithms for grouping trajectories points of moving objects. The motivation behind applying these algorithms is the addition of semantic information to spatiotemporal data. From the results obtained in many different fields of science, the proposed toolkit showed to be flexible and robust, in addition to being able to be easily applied in many problems, once it is properly configured.

Keywords: software engineering, bio-inspired artificial intelligence, optimization.

Capítulo

1

Introdução

O presente trabalho visa contribuir com o desenvolvimento de uma ferramenta de otimização com algoritmos genéticos (AGs) e também com um framework de software que facilite o uso de operações que envolvam esta técnica. A partir de uma configuração descrita pelo o usuário, a ferramenta deve ser capaz de resolver problemas nos quais os AGs possam ser aplicados. Apresentamos neste capítulo, as questões que motivaram a realização deste trabalho, seus objetivos, justificativas, metodologia e como o restante do documento está estruturado.

1.1 Motivação

Problemas de otimização ocorrem com freqüência em vários campos da ciência. Otimizar uma função matemática corresponde à procura do valor máximo ou mínimo que essa função pode assumir. Essas funções podem também possuir uma série de restrições para as variáveis a serem otimizadas. Problemas mais complexos, como os de controlar a injeção de combustível em um motor para maximizar o seu rendimento ou o de como cortar um bloco de madeira para minimizar suas perdas, também são exemplos de problemas de otimização.

Como solução para tais problemas, podemos citar os algoritmos evolucionários (AEs), que são definidos como um conjunto de métodos de otimização probabilísticos baseados na teoria da evolução das espécies de Charles Darwin (Darwin, 1859). Dentre as técnicas evolutivas, os AGs são o maior grupo de métodos que representam a aplicação de ferramentas evolutivas (Sivanandam & Deepa, 2008). Eles se baseiam na utilização de metodologias de seleção de indivíduos, operadores de *crossover* e de mutação. A substituição das soluções é feita geralmente por gerações de novos indivíduos. A metáfora dos comportamentos das espécies observadas por Darwin podem

ser simulados computacionalmente com o objetivo de obter valores ótimos para o problema a ser resolvido.

Apesar desses algoritmos terem sido propostos e validados, poucos esforços foram empregados na tentativa da criação de implementações capazes de, com apenas poucas alterações, poderem ser reusadas em problemas diversos. Para que estas implementações de AGs sejam reusadas em problemas distintos, muitas modificações precisam ser feitas no núcleo do código da implementação, tornando esta tarefa complexa e muito dependente da experiência do programador.

Como solução para agilizar a realização de uma otimização, pode-se fazer a utilização de um framework. De acordo com Johnson (1997), os frameworks são técnicas orientadas a objetos de reuso de software. Eles podem ser definidos sob duas perspectivas complementares. Na perspectiva de estrutura, os frameworks são vistos como um projeto que pode ser reusado como um todo ou em partes por um novo sistema. Na perspectiva do seu propósito, um framework é o esqueleto de uma aplicação que pode ser customizada por um desenvolvedor de aplicações. A chave para este conceito é a de que o desenvolvedor de software não precisa saber como o componente do framework está implementado, mas apenas saber como reutilizá-lo de acordo com a sua finalidade.

A criação de um framework para otimização com AGs se deve ao fato de existirem muitas aplicações com AEs e AGs em diversas áreas da ciência (Celeste *et al.*, 2004; Santos *et al.*, 2003; Hrstka & Kucerová, 2004; Thompson *et al.*, 1999; Santos *et al.*, 2007). Entretanto, a maioria desses trabalhos não utiliza uma ferramenta de otimização com algoritmos implementados ou não propõe frameworks para otimização, tornando o reuso de código da aplicação algo bastante complexo. Christiani (2009) e Paperin & Michalas (2004) explicitam a necessidade da criação de ferramentas que auxiliem a realização de otimizações com AEs e AGs. Podemos citar como iniciativas os frameworks JGAP (2002), Creation (Christiani, 2009) e JG2A (Bernal et al, 2009), e a ferramenta JAGA (2004). Elas possuem características e vantagens bastante relevantes, entretanto, apresentam limitações (i.e. não serem frameworks para otimização com AGs ou não possuírem implementações de algoritmos) que serão apresentadas na Seção 2.3.2. A ferramenta aqui proposta, procura suprir algumas dessas deficiências apresentando tanto um framework para otimização com AGs quanto uma ferramenta com AGs prontos para utilização.

1.2 Objetivo

O objetivo geral deste trabalho é criar uma ferramenta de otimização com AGs de propósito geral, capaz de ser utilizada em diversos problemas com pequenas ou nenhuma necessidade de alterações, baseada em um framework capaz de auxiliar na criação de AGs de forma ágil e simples. A ferramenta a ser construída, denominada de X-GAT, tem como objetivos específicos:

- Objetivo 1 Utilizar técnicas de engenharia de software para modelar a ferramenta de otimização de propósito geral e plataformas abertas na implementação, para que o produto final também seja aberto e de código livre para a comunidade científica;
- Objetivo 2 Projetar e implementar um esquema de configuração para ferramenta, dando assim alternativas ao usuário, para resolver o problema de otimização a ser solucionado;
- Objetivo 3 Construir a ferramenta visando sua expansibilidade, preocupando-se com a criação de um framework extensível, fazendo assim com que a adição de novas funcionalidades não seja uma tarefa árdua;
- Objetivo 4 Permitir que a ferramenta seja usada em plataformas diferentes, tanto com relação a sistemas operacionais quanto a linguagem de programação adotada na solução do problema;
- Objetivo 5 Aplicar a solução em diversos problemas de otimização contínua, mostrando em cada situação como a ferramenta foi utilizada e configurada para solucioná-los assim como também os resultados obtidos.

1.3 Justificativa

Vimos na Seção 1.1 que problemas de otimização são comuns em diversos campos da ciência e que poucos esforços foram feitos na tentativa de construção de ferramentas de otimização de caráter genérico. No que diz respeito a implementações de AGs de propósito geral, a variedade de opções para escolha de ferramentas de propósito geral é ainda menor, principalmente ferramentas de código aberto. Então, a construção de uma ferramenta que faça com que o usuário se preocupe apenas em conhecer a técnica, sem se preocupar em como implementá-la, facilitaria e agilizaria a otimização de um problema qualquer. O trabalho do usuário se resumiria em apenas configurar a

ferramenta, selecionando os tipos de operações e de dados que se adéqüem melhor ao problema a ser otimizado.

Ao alcançarmos os objetivos específicos enumerados na Seção 1.1, estaremos beneficiando os usuários da ferramenta X-GAT, uma vez que, para cada um deles temos:

Justificativa 1	A utilização de técnicas de engenharia de software para modelar a ferramenta ajudam a construir uma ferramenta efetiva, ou seja, que seja capaz de resolver um problema de otimização com AGs. O uso de plataformas abertas tem o objetivo de agilizar a construção da ferramenta, bem como receber futuramente um feedback dos usuários e adições de novas funcionalidades para as próximas versões da ferramenta;
Justificativa 2	Os AGs vêm sendo desenvolvidos há bastante tempo (Holland, 1975), então muitas melhorias em certos trechos do algoritmo já foram propostas e testadas em diversas situações. Permitir que o usuário tenha a possibilidade de selecionar operações distintas para diversos trechos do algoritmo é muito importante quando se tem o objetivo de fazer uma otimização de forma eficiente. Outra questão diz respeito ao domínio de cada problema de otimização, pois em certos domínios a seleção de cada funcionalidade influencia nos resultados obtidos, tanto com relação ao tempo de processamento quanto à precisão da solução obtida;
Justificativa 3	Seguindo a linha de raciocínio da justificativa anterior, de que os AGs e os AEs vêm sendo desenvolvido há bastante tempo, faz-se necessário também se preocupar com a expansibilidade da ferramenta, dado que tanto novas quanto outras operações não cobertas a princípio possam ser adicionadas em futuras versões da ferramenta;
Justificativa 4	Muitos dos problemas de otimização já resolvidos no passado ou até mesmo os atuais podem utilizar as mais distintas plataformas disponíveis no mercado. Para que a ferramenta aqui proposta possa ser aplicada nestes

	ambientes, torna-se necessária a sua implementação de forma a atender a este objetivo;
Justificativa 5	Para validar a ferramenta, mostrando que esta atendeu aos objetivos propostos nesta seção, é necessário aplicá-la em problemas distintos, mostrando como o processo para utilizá-la foi realizado. Além disso, a aplicação desta ferramenta em diversos problemas ajudará a fazer possíveis correções na implementação, bem como correções nas decisões de projeto de concepção.

1.4 Metodologia

A metodologia adotada no desenvolvimento desta ferramenta começa com uma revisão bibliográfica a respeito da problemática a ser solucionada e de algumas soluções que se propõem a resolvê-la. Depois de levantadas algumas questões que são necessárias ao desenvolvimento da solução aqui proposta, são mostradas as técnicas de engenharia de software utilizadas no processo de modelagem e construção da ferramenta X-GAT. Por último, a ferramenta é validada a partir da resolução de alguns problemas de otimização.

No decorrer desta subseção, serão mostradas as tecnologias utilizadas na implementação do X-GAT (Seção 1.4.1), o processo de desenvolvimento utilizado na sua construção (Seção 1.4.2) e os princípios que guiaram o seu desenvolvimento (Seção 1.4.3).

1.4.1 Tecnologias Fundamentais Adotadas

Para alcançarmos os objetivos deste trabalho, adotamos um conjunto de tecnologias e ferramentas que serão apresentadas a seguir. Algumas das tecnologias fundamentais para a solução do problema são: Java, XML e padrões de projeto.

Escolhemos a linguagem Java por diversos motivos. Java é uma linguagem orientada a objetos que possui mecanismos de garantia de robustez de software tais como: tratamento de exceções, *garbage collection* e ausência de aritmética de ponteiros (Arnold *et al.*, 2005). Outra característica importante é a portabilidade dos programas desenvolvidos nesta linguagem de programação, necessária então para cumprir o Objetivo 4.

A linguagem XML foi projetada para publicação de informações em uma representação flexível e extensível (Graves, 2003). Nesse contexto, esta linguagem de descrição de dados possibilita que o X-GAT receba os dados sobre a configuração desejada para execução (entrada), bem como que a sua saída resulte em dados que possam ser lidos e interpretados por qualquer outro sistema capaz de processar XML.

De acordo com Alexander (1977), um padrão de projeto descreve um problema que acontece várias vezes em certos tipos de situação. Dessa forma, ele deve descrever o núcleo da solução para tal problema, de forma que essa solução possa ser utilizada sistematicamente em diversas situações. Os padrões de projeto fazem com que se torne mais fácil o reuso de arquiteturas já consolidadas, assim como melhoram a documentação e manutenção de sistemas que já existem. Isso se deve ao fato de que os padrões de projeto fornecem uma definição explícita de interações de classe e objeto e sua intenção subjacente. Escolher padrões de projeto adequados para o desenvolvimento da ferramenta é no mínimo interessante, pois estes fornecem soluções testadas e aprovadas.

1.4.2 Processo de Desenvolvimento

O processo de desenvolvimento adotado foi o evolucionário (Sommerville, 2007). Nesse processo, as atividades de especificação, desenvolvimento e validação são intercaladas. A vantagem da utilização de um processo de software com abordagem evolucionária para este problema é a geração de um protótipo a ser validado. Este protótipo vai sendo ajustado até atender à especificação final. Ao final de vários ciclos é esperado que se obtenha um software que atenda aos requisitos levantados no início do projeto.

Cada operação da ferramenta X-GAT foi testada e validada primeiramente de forma individual. Após o processo de validação das operações unitárias, foram criadas algumas implementações de AGs a fim de validar que o conjunto de operações criadas daria suporte suficiente para que a heurística de busca funcionasse de forma correta, gerando assim os primeiros protótipos. A partir de diversos experimentos realizados (mostrados no capítulo 4) para fins de validação, a ferramenta foi empregada em diversas áreas, com o objetivo de mostrar o seu potencial de aplicação.

1.4.3 Princípios

Alguns princípios guiaram o desenvolvimento desta ferramenta. Eles podem ser considerados também como pré-requisitos necessários para o desenvolvimento do X-GAT. Os princípios seguidos foram:

- **Tempo aceitável de execução:** o software deve atingir um tempo razoavelmente aceitável quando realizar uma otimização;
- **Qualidade de resposta**: desde que a ferramenta seja configurada adequadamente, a resposta deve sempre se aproximar do ótimo, assim como propõe os AEs e os AGs;
- **Extensibilidade**: a ferramenta deve ser construída visando possíveis extensões e/ou especializações para qualquer área da ciência, fornecendo apenas mecanismos básicos que possam ser estendidos;
- **Escalabilidade**: a implementação deve ser capaz de resolver tanto problemas triviais como problemas de complexidade mais elevada;
- **Fácil manutenção**: o produto gerado deve ser fácil de manter, com uma documentação completa e detalhada sobre o que está disponível para uso;
- **Simplicidade de uso:** desde que o usuário conheça a técnica, não é necessário que este conheça o núcleo do código da aplicação, mas que ele apenas saiba quais operações podem ser utilizadas para realizar uma otimização. Desta forma, o usuário estará programando de forma declarativa, apenas invocando as operações desta ferramenta, sendo desta forma todo o núcleo do código abstraído.

1.5 Estrutura do trabalho

O trabalho está estruturado de acordo com a seguinte lista de capítulos:

- No segundo capítulo, “Fundamentação Teórica”, são mostrados os conceitos e as terminologias envolvidas para a construção de AEs e AGs. São mostrados ainda trabalhos relevantes e relacionados ao tema abordado, além de ferramentas que se assemelham com o X-GAT, com um comparativo entre elas;

- No terceiro capítulo, “A Ferramenta X-GAT”, é apresentado como foi concebida a ferramenta, atendendo as necessidades levantadas no segundo capítulo;
- No quarto capítulo, “Testes e Resultados”, é apresentado um conjunto de testes em áreas distintas, mensurando, para cada aplicação, o grau de qualidade das soluções obtidas;
- No quinto e último capítulo, a dissertação é concluída com discussões sobre os resultados obtidos do trabalho, as contribuições alcançadas e mostra também algumas direções para trabalhos futuros.

Capítulo

2

Fundamentação Teórica

O capítulo 2 tem o objetivo de fundamentar a técnica de otimização de AGs a partir de diversos conceitos, discussões sobre a sua aplicabilidade em diversos domínios e como se encontra o estado da arte de ferramentas que utilizam essa técnica de otimização. O capítulo começa com uma apresentação dos conceitos e terminologias encontrados na literatura que tratam dos AEs e dos AGs (Seções 2.1 e 2.2, respectivamente). Em seguida, trazemos na Seção 2.3, os trabalhos relacionados são discutidos e esta seção está organizada do seguinte modo: a subseção 2.3.1, que apresenta trabalhos que reforçam a necessidade de criação de ferramentas para otimização de forma genérica; e a subseção 2.3.2, que discute soluções que se assemelham à ferramenta aqui proposta. Por fim, fazemos as considerações finais do capítulo na Seção 2.4.

2.1 Os Algoritmos Evolucionários

Antes de começarmos a definir os AEs, discorreremos um pouco sobre a teoria da evolução. Em meados de 1850, o cientista Charles Darwin percebeu por meio de de sua pesquisa que existiam pequenas diferenças entre animais da mesma espécie e que viviam em ambientes diferentes, sendo cada um destes mais bem adaptados ao ambiente no qual eles viviam. Essas observações foram explicadas no livro “A origem das espécies”, de 1859, criando assim a teoria da evolução natural das espécies, indo de encontro à teoria do criacionismo.

Segundo a teoria evolucionista, os indivíduos de um mesmo ambiente entram em disputa pelos recursos presentes naquele local. Os indivíduos da mesma espécie que não conseguem obter uma maior quantidade destes recursos tendem a gerar menos descendentes, ou seja, os indivíduos mais fortes e aptos ao ambiente possuem uma maior

probabilidade de propagar suas características genéticas para os seus descendentes. Esse processo é denominado de seleção natural ou também pode ser encontrado na literatura como “a sobrevivência do mais forte”.

Combinando as características de dois indivíduos, pode-se obter um indivíduo mais bem adaptado ao ambiente. Entretanto, nem sempre isso é verdade, pois a troca de informações genéticas entre estes dois indivíduos também pode gerar um indivíduo menos adaptado ao ambiente em que eles competem. Essa afirmativa é derivada do mecanismo de transmissão da informação genética, que pode ser mais bem detalhada em livros de biologia, não sendo interessante então nos aprofundarmos na questão.

Tendo definido algumas questões da teoria da evolução, podemos agora entrar em alguns detalhes relacionados aos AEs. De acordo com Linden (2008), os AEs usam modelos computacionais de processos naturais de evolução como uma ferramenta para resolver problemas. Em Michalewicz (1999), os AEs são definidos como um conjunto de métodos probabilísticos baseados na teoria da evolução de Charles Darwin e surgiram para lidar com problemas complexos de otimização. De acordo com ambos, o comportamento das espécies observado por Darwin pode ser simulado computacionalmente com o objetivo de obter valores otimizados para o problema.

Os AEs podem ser ainda subdivididos em quatro classes (Sivanandan & Deepa, 2008): Algoritmos Genéticos (AGs), Programação Genética (PG), Estratégias Evolucionárias (EEs), Programação Evolucionária (PE). A diferença básica entre estas classes diz respeito à natureza dos esquemas de representação de soluções, aos operadores de reprodução e aos métodos de seleção.

Christiani (2009) define essas quatro classes da seguinte maneira:

- **Programação Genética:** o foco recai sobre o desenvolvimento de programas de computador válidos. A população consiste em árvores de análise de programas de computador que podem ser executados e que um valor de aptidão pode ser dado a cada programa;
- **Programação Evolucionária:** semelhante à programação genética, a meta é a evolução dos programas de computador. A diferença reside no fato de que a estrutura do programa é fixa e somente alguns parâmetros numéricos são evoluídos;
- **Estratégias Evolucionárias:** focam diretamente em otimizar um ou mais parâmetros de valor real. O cromossomo pode ser visto como um

vetor *n*-dimensional que representa uma possível solução. Estes então evoluem por meio de um operador de mutação;

- **Algoritmos Genéticos:** esta categoria tem semelhanças com as estratégias evolucionárias, a solução é codificada discretamente no genótipo (muitas vezes como bits) ao invés de vetores de números reais. Os operadores de reprodução aplicados são o *crossover* e a mutação.

2.2 Os Algoritmos Genéticos

Dentre as técnicas de AEs, os AGs são os mais conhecidos. Os AGs são métodos de otimização baseados na teoria da evolução, apresentada por Charles Darwin em 1859 no livro "A Origem das Espécies" (Darwin, 1859). Isso significa que, utilizando tal teoria, quanto mais um indivíduo consegue se adaptar ao ambiente, maior é a probabilidade deste sobreviver e gerar descendentes. O que caracteriza realmente um AG, diferentemente das outras três técnicas dos AEs, é o fato dele utilizar o conceito do processo de evolução natural combinado com os operadores de reprodução *crossover* e mutação e um operador de seleção. O primeiro trabalho com AGs foi realizado por Holland (1975). Neste trabalho o autor mostra como aplicar os princípios de evolução natural descritos por Darwin para construir um AG.

Apesar dos AGs possuírem na composição do seu nome o termo *algoritmo*, estes são na verdade heurísticas de busca para ótimos locais, não garantindo assim que o valor ótimo para a solução seja obtido ao final de sua execução.

De acordo com Michalewicz (1999), um AG deve conter:

- Uma representação genética para as soluções;
- Uma forma de criar a população inicial de soluções;
- Uma função de avaliação para que seja atribuída uma aptidão a cada solução;
- Operadores genéticos que alterem a composição genética dos descendentes gerados;
- Valores para os vários parâmetros usados nos AGs (tamanho da população, taxa de *crossover*, taxa de mutação, etc).

Nas próximas subseções, detalharemos algumas questões em relação aos AGs para que seja levantado um conjunto de informações com o propósito de gerar uma arquitetura que atenda aos objetivos definidos para a ferramenta X-GAT.

2.2.1 Terminologia e um Algoritmo Genético Básico

Para modelarmos nossa implementação adotaremos a terminologia apresentada na Quadro 1.

Quadro 1 – Terminologia adotada.

Terminologia	**Significado**
Gene	Representa uma parte de uma solução. Um gene representa uma das variáveis de entrada do problema a ser otimizado.
Cromossomo ou Indivíduo	É um conjunto de genes que compõem uma possível solução do problema.
População	Conjunto de indivíduos que competem pelos recursos de um ambiente.
Geração	É definida como uma população em certo tempo *t*.
Nota de aptidão (*fitness*)	É um valor atribuído a um indivíduo. Esse valor tem o objetivo de mensurar o nível de adaptação do indivíduo ao ambiente. Dessa forma, os indivíduos podem ser comparados uns com os outros em uma determinada geração.

Com as definições do Quadro 1, podemos agora mostrar um AG básico. Cada parâmetro de entrada de um problema de otimização é visto como um gene. Os indivíduos são um conjunto de genes, representando assim uma solução possível para o problema de otimização. Um conjunto de soluções (i.e conjunto de indivíduos) é denominado população. Cada população vive em um ambiente em certo tempo. Uma população que vive em um tempo ($t = 0$) é evoluída através da implementação do AG e gera a próxima geração de indivíduos ($t = 1$). Cada solução é avaliada gerando um valor de aptidão que é usado para selecionar os melhores indivíduos, ou seja, os indivíduos com os melhores valores de nota de aptidão possuem uma maior probabilidade de se reproduzir e gerar descendentes.

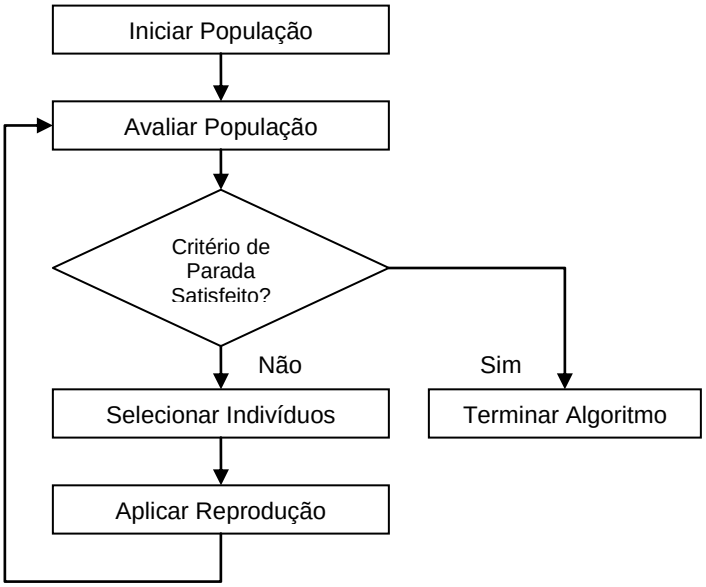


Figura 1 – Um Algoritmo Genético Básico.

Como pode ser visto na Figura 1, um AG deve gerar inicialmente uma população de indivíduos, e depois, essa primeira população de indivíduos é analisada. A aptidão de cada indivíduo ao ambiente é determinada, ou seja, o quanto uma solução proposta é relativamente melhor quando comparada com as outras soluções. Assim como na teoria de Darwin, os indivíduos mais fortes (com um valor de nota de aptidão melhor) possuem uma maior chance de serem selecionados para gerarem descendentes. Enquanto a condição de parada não for atingida, o algoritmo cria novas populações de indivíduos. A partir da função *Aplicar Reprodução*, as técnicas de *crossover* e mutação são aplicadas nos indivíduos selecionados da população.

As operações de reprodução (*crossover* e mutação) de um AG possuem uma taxa de ocorrência que deve ser levada em consideração antes de serem aplicadas. Quando essas operações são utilizadas na execução de uma otimização, é gerado um valor aleatório entre 0 e 100, que é comparado com essa taxa. Se esse valor for menor ou igual a essa taxa, então tal operação de reprodução é executada. Entretanto, se esse número for maior que essa taxa, a operação de reprodução não é aplicada no indivíduo. Por exemplo, caso se deseje usar uma taxa de 80% de *crossover* em uma determinada otimização, será sorteado um valor de 0 a 100. Caso este valor seja menor ou igual a 80, o *crossover* é aplicado nos dois indivíduos que estão se reproduzindo. Caso este número seja superior a 80, a operação de *crossover* não acontecerá. Assim como na natureza, as

taxas de *crossover* devem ser sempre maiores do que as taxas de mutação. As taxas de mutação são geralmente baixas, com o objetivo de seguir fielmente a sua taxa de ocorrência real nos processos de reprodução biológica. De Jong (1975) realizou um estudo aprofundado sobre o efeito da variação das taxas de mutação e de *crossover*. De Jong verificou que a melhor opção, na maioria dos casos, consiste em escolher uma probabilidade de mutação igual ao inverso do tamanho da população, ou seja, para uma população de 100 indivíduos essa taxa deve ser de (1/100 – 1%). Sobre o efeito da probabilidade de *crossover,* o autor realizou um estudo envolvendo as probabilidades de 80%, 60% e 40% para uma população de 50 indivíduos. Neste estudo, os melhores resultados foram obtidos com as taxas de 80% e 60% de ocorrência de *crossover*.

Em muitas aplicações que utilizam AGs, a forma mais comum de codificação é a utilização de strings binárias de comprimento fixo. A razão para esta utilização é que, na teoria em que os AGs foram criados, esta representação foi utilizada. Esta forma de codificação será usada para as definições e exemplos das operações de *crossover* e mutação. Neste tipo de codificação, os parâmetros de entrada a serem otimizados são analisados na base binária, com o objetivo de simular computacionalmente o processo de reprodução que acontece na natureza. Cada bit de um número representa uma parte de um cromossomo na natureza. Na natureza, pedaços dos cromossomos de dois pais são trocados durante o processo de reprodução. Para que este processo seja simulado computacionalmente, são trocados bits entre dois indivíduos, fazendo com que novos indivíduos sejam gerados a partir dessas pequenas partes oriundas dos pais.

Na operação de *crossover*, partes do número (bits) são trocados entre dois indivíduos gerando um descendente diferente dos originais. A Figura 2 mostra como essa operação é realizada. Primeiro, certa quantidade de bits é escolhida para serem trocados entre as soluções. No exemplo da Figura 2, quatro bits foram escolhidos para serem trocados entre os indivíduos. Assim como na genética, as partes trocadas geram dois indivíduos diferentes dos originais.

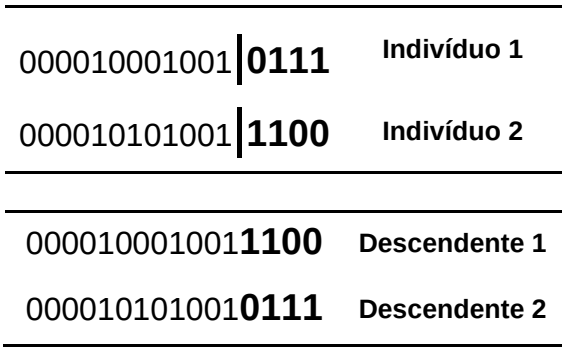


Figura 2 – Operação de *crossover* binário.

Após a aplicação da operação de *crossover*, é aplicada aos novos indivíduos gerados a operação de mutação. Na operação de mutação, cada bit do descendente gerado pelo *crossover* possui uma pequena probabilidade de ser mudado. Isso significa que se a mutação ocorrer em um bit, um valor que era 0 (zero) se transformará em 1 (um) e vice-versa. A Figura 3 mostra como essa operação é executada. Essa operação é aplicada a cada bit do descendente gerado pelo *crossover* desde que seja ativada para execução.

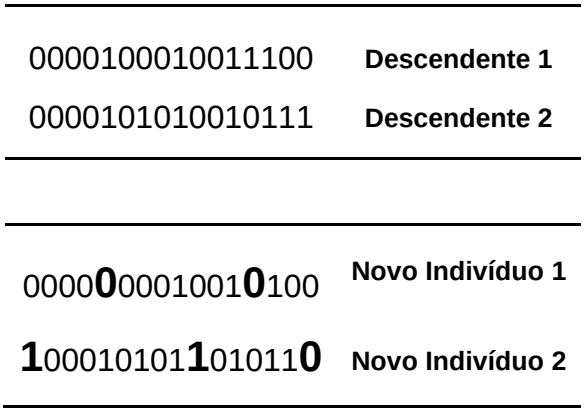


Figura 3 – Operação de mutação binária.

A utilização da representação binária, como pode ser vista acima, é muito simples para que a teoria dos AGs seja analisada. Entretanto, se o problema a ser resolvido possui parâmetros contínuos e o usuário procura uma boa precisão numérica, essa abordagem possui alguns problemas. Para adicionar mais um dígito decimal na precisão, é necessária a adição de 3.3 bits (Michalwicz, 1999) na solução. Então, a convergência do algoritmo começa também a perder velocidade, tornando-se muito lenta.

Para resolver esse tipo de problema, a codificação de números de ponto flutuante pode ser utilizada. Essa abordagem é mais legível se comparada à abordagem

de codificação binária. Cada gene é representado como um número real (34.15, por exemplo). Alguns experimentos comparando as abordagens binária e de ponto flutuante são mostrados em Janikow & Michalewicz (1991), Bramlette & Cusic (1989), Bramlette (1991) e Furuya & Haftka (1993). Em todos esses trabalhos, foi mostrado que a representação de ponto flutuante obtém melhor desempenho que a representação binária. As representações não-binárias sempre foram consideradas uma escolha razoável para problemas nos quais o tamanho do espaço de busca ou as características do problema são naturalmente representadas por números ou letras. A partir destes resultados, foi decido que não serão implementados no X-GAT, os tipos binários para manipulação dos operadores genéticos.

2.2.2 Porque Utilizar os AGs

De acordo com Linden (2008), os AGs são uma técnica de busca com as seguintes características:

- **Paralela:** mantém uma população de soluções que são simultaneamente avaliadas;
- **Global:** AGs não utilizam apenas informação local, logo não ficam presos necessariamente em ótimos locais;
- **Não totalmente aleatória:** possuem alguns componentes aleatórios, mas usam a informação corrente da população para determinar o próximo estado da busca, sendo assim apenas parcialmente aleatória.
- **Não é afetada por descontinuidades na função ou em suas derivadas:** AGs não usam informações de derivadas na sua evolução nem necessitam de informações sobre o seu entorno para poder efetuar sua busca.
- **Capaz de lidar com funções discretas e contínuas:** AGs podem lidar com funções reais, discretas, booleanas e até mesmo não numéricas.

Complementando as vantagens mencionadas acima, Sivanandan & Deepa (2008) citam as seguintes vantagens da utilização de AGs:

- **Simplicidade conceitual:** conceitualmente, os AGs são simples, pois o algoritmo consiste em apenas fazer uma inicialização, variação (com alterações) iterativa e uma forma de selecionar os melhores para cruzamento de informações;

- **Larga aplicabilidade:** eles podem ser aplicados em quaisquer problemas que possam ser formulados por meio de uma função de otimização (utilizada como função de aptidão no AG);
- **Hibridização com outros métodos:** podem ser combinados com técnicas de otimização mais tradicionais;
- **Paralelismo computacional:** geralmente as soluções (indivíduos) podem ser analisadas independentemente umas das outras, sendo necessária apenas uma fase de troca de dados entre eles, deixando assim margem para que esse processamento seja paralelizável;
- **Robusto para mudanças dinâmicas:** geralmente, os métodos tradicionais não são robustos para mudanças repentinas no ambiente em que eles estão sendo executados. Ao contrário destas técnicas tradicionais, os AGs podem ser usados para adaptar as soluções para ambientes dinâmicos;
- **Resolvem problemas que não possuem solução conhecida:** uma outra vantagem desta técnica inclui a habilidade para resolver problemas para os quais não existem soluções exatas e conhecidas pelo homem.

Entretanto, quando se conhece um algoritmo que seja capaz encontrar a solução ótima do problema, não se recomenda aplicar uma heurística de busca como os AGs, já que estes não garantem que o ótimo global será encontrado.

2.2.3 Outras Questões sobre AGs

Muitos autores discutem questões interessantes a respeito da implementação de AGs. Discutiremos aqui duas destas questões: a utilização do processo de elitismo; e a detecção de convergência primária das soluções encontradas pelo AG.

De acordo com Linden (2008), a técnica de elitismo é uma pequena modificação que não altera o tempo de processamento de um AG, mas que garante que o seu desempenho cresça no decorrer da evolução das populações administradas. Na idéia da execução de um AG com elitismo, certa quantidade dos melhores indivíduos de uma geração (t) devem ser colocados na geração (t+1) visando garantir que as características destes melhores indivíduos estejam presentes nas gerações futuras. Isso garante que a geração posterior será pelo menos igual ou melhor que a geração anterior. A implementação desta modificação, apesar de muito simples, colabora bastante para o desempenho de uma otimização com AGs, pois haverá garantia de que o melhor

indivíduo encontrado durante a execução da heurística não será eliminado das populações evoluídas. Para a ferramenta X-GAT, em qualquer um dos AGs implementados, o usuário tem a possibilidade de habilitar a opção de utilizar o elitismo em uma otimização.

Quando apenas o melhor valor encontrado (um único indivíduo) pela execução de um AG tem a garantia de estar na geração seguinte, esta técnica não insere problemas. Entretanto, considerando que um conjunto de melhores indivíduos encontrados vão perdurar em uma geração seguinte insere o problema de convergência prematura de soluções, onde uma área específica do espaço de busca irá ser mais explorada em detrimento de outras áreas do espaço de busca.

Determinar que, durante a execução de um AG, o valor do ótimo global encontrado permaneça o mesmo durante sucessivas evoluções, não deve ser um processo que se baseie na nota de aptidão, pois indivíduos que possuem características bem diferentes podem ter notas de aptidão bastante parecidas. A detecção da convergência primária (ou convergência prematura) deve ser realizada com base na comparação entre os genes (variáveis a serem otimizadas do problema) dos indivíduos que compõe a geração que está sendo evoluída. Se a determinação da convergência primária fosse baseada apenas na nota de aptidão dos indivíduos, poderia acontecer de existir um genótipo muito diferente na população e ser inferido que este tipo de convergência ocorreu. Na ferramenta X-GAT, esta metodologia é disponibilizada para utilização através do critério de parada *Convergence*, que pode ser encontrada na Seção 3.4.6.

Outra questão importante e que deve ser levada em consideração é a dificuldade da representação de uma solução como um cromossomo. A natureza do problema ao qual o AG vai ser aplicado pode gerar uma série de cromossomos inválidos, sendo assim necessário que haja muito cuidado no momento dessa modelagem cromossômica. Os tipos de dados de cada gene devem ser escolhidos com cuidado, para que não sejam gerados descendentes fora do espaço de busca.

2.3 Trabalhos Relacionados

Os trabalhos relacionados foram subdivididos em duas categorias: aplicações de AGs em diversas áreas; e ferramentas para otimização com AGs. Na Seção 2.3.1, serão mostradas várias aplicações de AEs e AGs em diversas áreas da ciência, dando ênfase a necessidade de criação de ferramentas que facilitem sua aplicação em vários problemas de otimização. Em seguida, na Seção 2.3.2 serão mostradas algumas ferramentas que

visam resolver esta problemática, sendo levantadas as características relevantes de cada trabalho.

2.3.1 Aplicações de AGs em Diversas Áreas

Os AGs podem ser aplicados a diversos tipos de problemas. Esta técnica tem sido aplicada em problemas na área das engenharias. Serão mostrados primeiramente alguns trabalhos em áreas gerais da engenharia, e em seguida, algumas aplicações de AGs em recursos hídricos, já que esta foi uma área na qual se aplicou o X-GAT na seção de testes e resultados (Capítulo 4). Por último, serão mostradas aplicações de AGs em algumas outras áreas, mostrando o quão amplo é a aplicabilidade desta técnica.

No trabalho de Matou *et al.* (2000), são mostrados diversos tópicos comumente achados e discutidos sobre os AEs e alguns testes interessantes a respeito de diversas configurações de algumas dessas técnicas. Os algoritmos são aplicados com o objetivo de otimizar o custo da estrutura de vigas mistas.

Ainda sobre este mesmo tópico, otimização de custo de vigas mistas, em Senouci & Al-Ansari (2009), são mostrados resultados interessantes da aplicação de um AG neste tipo de otimização com custo. No problema em questão, são otimizadas cinco variáveis. Uma característica interessante desta aplicação é que o AG clássico é estendido com o objetivo de tornar a busca pelo ótimo mais efetiva. O núcleo dos AGs, como as operações de *crossover*, mutação e seleção são mantidas. Dois exemplos e um estudo de caso foram utilizados para ilustrar o modelo desenvolvido em gerar as soluções de projetos de vigas que conseguiram o melhor custo mínimo total. Esse novo recurso deve ser útil para projetistas estruturais e é interessante para o avanço das práticas de projetos existentes de vigas mistas.

Algumas melhorias nos operadores dos AGs para prevenção de convergência primária podem ser vistas em Hrstka & Kucerová (2004). Nesse trabalho, os autores avaliam diversos AEs e focam principalmente na resolução de problemas multi-objetivo. Como critério de avaliação do método, ou seja, para de calcular a nota de aptidão, foram testadas diversas funções matemáticas, e o algoritmo implementado se comportou de forma confiável na resolução destes problemas.

Em Kisi & Guven (2010), são mostrados resultados da aplicação de um AE, denominado programação genética linear, na estimativa correta da concentração de sedimentos transportados por um rio. Os resultados deste trabalho mostram que este AE obteve um desempenho melhor do que algumas técnicas bastante utilizadas atualmente, como redes neuro-fuzzy e redes neurais.

Em Diniz (1999), é mostrada a execução de um processo de calibração automática de parâmetros de entrada de modelos hidrológicos chuva-vazão. Neste trabalho, o AG SCE-UA (Shuffled Complex Evolution) é utilizado para calibrar um modelo hidrológico com 10 parâmetros de entrada. O experimento é realizado com dados da bacia hidrográfica do rio Mamuaba, localizado no litoral da Paraíba. A utilização de um AG no processo de calibração automática de parâmetros de entrada se mostrou mais interessante que a manual, pois esta exige grande conhecimento do modelo aplicado, bem como sobre a região hidrológica que está sendo estudada.

A dissertação de mestrado de Celeste (2002) mostra o desenvolvimento e a aplicação de um modelo de otimização que combina programação não linear e AGs para a operação em tempo real de um sistema de recursos hídricos. Esse sistema é composto por um reservatório sujeito a múltiplos usos e um conjunto de poços, responsáveis pelo abastecimento d'água da cidade de Matsuyama, situada na província de Ehime, no Japão. Os resultados mostraram que os AGs são heurísticas de busca eficientes e robustas para ajudar na otimização de sistemas de recursos hídricos com múltiplas finalidades.

Em Santos *et al.* (2003), é mostrada uma modificação no AG denominado SCE-UA, sendo esta implementação testada em algumas funções matemáticas de difícil resolução e posteriormente, em parâmetros de erosão presentes em um modelo físico hidrossedimentológico.

Em Santos *et al.* (2007), é mostrado um trabalho que objetiva a aplicação de métodos de extração e seleção de atributos para classificação de regiões geográficas é discutida. Atributos de textura e forma são extraídos das regiões em estudo e métodos de seleção são aplicados a fim de reduzir a dimensionalidade do espaço de atributos. Para realizar a classificação, é utilizada uma implementação de um AG objetivando a redução das dimensões de um mapa sem haver perda no poder discriminatório entre os dados.

No trabalho de Thompson *et al.* (1999), são investigadas algumas hipóteses sobre projetos eletrônicos de hardware. Questões de hardware e software são discutidas e também é mostrado que as técnicas evolucionárias podem explorar as formas e os processos que são naturais para o meio eletrônico e que também requisitos não-comportamentais podem ser integrados neste processo de modelagem, tais como tolerância a falhas.

Na dissertação de mestrado de Ávila (2002), o autor desenvolve uma nova metodologia para operadores genéticos, utilizando codificação de números reais,

objetivando melhorar a varredura do espaço de busca da solução ótima. A eficácia destes AGs e destes novos operadores genéticos é verificada pela sua aplicação em diversas funções teste de complexidade elevada. Em seguida, os novos métodos são aplicados em um problema eletromagnético, que determinou a conformação da superfície do refletor de uma antena refletora offset. O objetivo desta otimização é obter uma antena de satélite que produza um diagrama de radiação que cubra uniformemente o território brasileiro. Os resultados obtidos mostram os AGs como uma boa solução, eficiente e confiável, para a otimização de problemas complexos.

O trabalho de Souza (2004) mostra a implementação de um AG para auxiliar a tomada de decisão no que diz respeito a minimizar os custos com atividades relacionadas à colheita e ao transporte principal de madeira. As soluções com AGs apresentaram então custos de colheita, transporte e estoque excedente de madeira melhores dos que os resultados obtidos com métodos mais tradicionais de otimização.

Os trabalhos mencionados nesta subseção apenas reforçam a necessidade da criação de uma ferramenta capaz de auxiliar: (1) a criação de novos métodos de busca com AGs através do projeto de um framework com operações básicas; (2) a aplicação de AGs implementados e testados em diversos problemas de otimização, requerendo apenas alterações na configuração de uma ferramenta.

2.3.2 Ferramentas para Otimização com AGs

Muitas implementações de AGs são realizadas com propósitos específicos. Entretanto, também existem projetos que procuram ser genéricas o suficiente para resolverem qualquer tipo de problema no qual um AG pode ser utilizado. Esta subseção mostrará algumas tentativas de criação de frameworks para otimização com AGs de distribuição livre e de código aberto.

A ferramenta mais conhecida atualmente é o Java Genetic Algorithm Package (JGAP - http://jgap.sourceforge.net/). Esta ferramenta fornece um conjunto básico de mecanismos que podem ser usados para aplicações de princípios evolucionários em soluções de problemas. Os meios para construção de um AG clássico ou um Algoritmo Evolucionário são apenas disponibilizados, tornando necessária a configuração do problema e também a implementação do algoritmo a ser utilizado. Os dados de simulação podem ser importados ou exportados como um arquivo XML, tanto os dados de entrada quanto os de saída. Entretanto todo este procedimento deve ser feito pelo usuário, não existindo uma operação direta de importação e exportação de dados. As operações evolucionárias podem ser estendidas facilmente, tornando prática a

implementação de novas funcionalidades. Muitas funcionalidades estão disponíveis na ferramenta, fazendo dela uma ferramenta evolucionária bastante robusta. O JGAP também dá suporte à otimização utilizando *grids* computacionais. De acordo com Foster (2001), um *grid* computacional coordena recursos que não estão sujeitos a um controle centralizado, ou seja, um *grid* é um ambiente computacional no qual muitos computadores independentes executam tarefas distribuídas por um servidor mestre, podendo então uma otimização ser realizada de forma paralela por vários computadores.

Outra ferramenta é a Java GALib (http://java-galib.sourceforge.net/). Comparado ao JGAP é uma ferramenta menos robusta, entretanto existe a implementação do AG clássico que pode ser facilmente utilizada. A ferramenta só dá suporte à generalidade de aplicação, ela não é facilmente extensível e possui uma documentação bastante escassa.

A ferramenta Java API for Genetic Algorithms (JAGA - http://www.jaga.org/index.html) é outra solução que procura ser genérica. Essa ferramenta é descrita como um esforço colaborativo para computação evolutiva e sistemas de software de grupos de pesquisa de engenharia. A JAGA contém uma grande variedade de algoritmos evolucionários prontos para usar, e inclui os operadores genéticos e as representações genótipo.

A Quadro 2 resume o que foi mencionado a respeito das ferramentas disponíveis para otimização genérica com AGs. As seguintes características foram adotadas como critérios nesta análise:

- **Genérica***:* a ferramenta se propõe a resolver qualquer problema de otimização no qual um AG pode ser aplicado;
- **Configurável***:* a ferramenta possui diversas funcionalidades que podem ser modificadas pelo usuário;
- **Extensível**: A arquitetura da ferramenta pode ser facilmente estendida para que novas funcionalidades possam ser adicionadas;
- **Portabilidade de Sistema Operacional:** as tecnologias adotadas na implementação permitem que as instâncias da aplicação executem em qualquer sistema operacional;
- **Construção de Sistemas Heterogêneos:** a ferramenta de alguma forma permite que o usuário utilize outras linguagens de programação para construir a aplicação final;

- **Algoritmos prontos para executar:** além de fornecer um framework para facilitar a construção de AEs e AGs, a ferramenta já fornece algoritmos prontos para execução;
- **Documentação:** existe documentação além de comentários de código, mostrando como a ferramenta pode ser utilizada para resolução de problemas de otimização.

O JGAP mostrou ser uma ferramenta a mais completa, quando analisados os critérios da Quadro 2. Entretanto, a ferramenta requer do usuário certo conhecimento a respeito de AGs e de AEs para que possa ser utilizada de forma correta, já que nenhum algoritmo pode ser executado diretamente, necessitando assim que o usuário utilize um conjunto de operações para realizar uma simulação. A ferramenta Java GALib, apesar de ter sido considerada configurável, não possui muitas funcionalidades disponíveis. Outro problema é que ela não foi projetada para ser facilmente extensível, não dá suporte à construção de sistemas heterogêneos e não possui documentação. Já a ferramenta JAGA pode ser considerada, de certa forma, bastante completa, ficando apenas a desejar quanto a permitir a construção de sistemas heterogêneos e não possuir documentação nem de comentários de código. Uma característica importante que deve ser ressaltada a respeito do JAGA é que ele dá suporte a tipos de dados para representação de seqüências de proteínas, sendo esta característica bastante aplicável na resolução de problemas biológicos.

Quadro 2 - Comparação entre ferramentas disponíveis.

	Genérica	Configurável	Extensível	Portabilidade de Sistema Operacional	Construção de sistemas heterogêneos	Algoritmos prontos para executar	Documentação	Execução em *grids* computacionais
JGAP	O	O	O	O	O	X	O	O
Java GALib	O	O	X	O	X	O	X	X
JAGA	O	O	O	O	X	O	X	X

Legenda: O – A ferramenta contém a característica.

X – A ferramenta não contém a característica.

A ferramenta proposta neste trabalho visa preencher todas as colunas do Quadro 2, podendo dar ao usuário, um grande leque de funcionalidades sem este ter que se preocupar em fazer configurações complexas.

2.4 Considerações finais

O principal objetivo deste capítulo foi contextualizar a problemática a ser solucionada, bem como discutir algumas técnicas que podem ser utilizadas para resolvê-la. Foram mostradas algumas heurísticas de busca que fazem parte da literatura dos AEs e posteriormente, os AGs foram definidos e várias questões fundamentais para sua implementação foram levantadas. Em seguida, foram mostrados diversos trabalhos em áreas distintas que utilizam estas técnicas em algum processo de otimização. Por último, foram mostradas algumas ferramentas que se propõem a resolver o problema abordado nesta dissertação, sendo levantadas algumas lacunas a serem preenchidas pela ferramenta X-GAT.

Depois de definida em detalhes a heurística de busca a ser implementada (AGs) e terem sido levantadas diversas questões de implementação a serem levadas em consideração neste trabalho, é mostrado no próximo capítulo, como foi realizado o processo de modelagem e construção da ferramenta X-GAT.

Capítulo

3

A Ferramenta X-GAT

Este capítulo apresenta detalhes da concepção e implementação da ferramenta X-GAT. Primeiro, é realizado um levantamento das entidades que são necessárias para a criação da ferramenta (Seção 3.1). Em seguida, são mostrados os padrões de projeto (*design patterns*) utilizados na implementação da ferramenta (Seção 3.2). Após a definição das entidades e dos padrões de projeto a serem seguidos, são apresentados os AGs que foram implementados e validados pela ferramenta X-GAT (Seção 3.3). Na seqüência, são contextualizadas as funcionalidades (i.e. partes configuráveis de um AG) que podem ser utilizadas em uma otimização com AGs (Seção 3.4). Na ferramenta X-GAT, as funcionalidades a serem utilizadas no processo de otimização devem ser informadas ao AG, a partir de um arquivo XML. A Seção 3.5 define a posição e a forma como cada *tag* de cada funcionalidade neste arquivo XML deve ser escrita para a ferramenta X-GAT. Por último, são mostradas algumas considerações relevantes a respeito da construção da ferramenta.

3.1 Entidades Necessárias ao Desenvolvimento da Ferramenta X-GAT

Muitas implementações de AEs e AGs foram construídas desde que esta técnica foi criada. Uma grande parte dessas implementações utiliza o paradigma de programação imperativo (ou procedimental), porque uma área forte na qual esta técnica é aplicada é a área das engenharias, em que as linguagens de programação imperativas como Fortran ainda são largamente utilizadas. A ferramenta aqui apresentada, diferente desta tendência de utilização do paradigma imperativo, utiliza o paradigma Orientado a Objetos (OO) na sua concepção e implementação.

O paradigma OO foi criado para modelar conceitos do mundo real a partir da sua representação como objetos. Os conceitos centrais deste paradigma são: objetos,

herança, classes e subclasses (Watt, 2003). Um objeto é um elemento computacional que representa alguma entidade no domínio da solução. Objetos similares são agrupados em classes. Uma subclasse é uma classe que herda as características (estrutura e comportamento) de uma ou mais classes, podendo ser estendida ou ter o comportamento herdado modificado. Subclasses são definidas por meio da criação de relacionamentos de generalização/especialização entre classes e que possui a propriedade de herança.

Para a construção de diagramas que representem conceitos do sistema, atividades e interações com o usuário, será adotado daqui em diante a Unified Modelling Language (UML). Classes, objetos e também eventos externos ao sistema (ação do usuário, por exemplo) podem ser representadas nesta linguagem (Larman, 2007).

A primeira fase para a construção de um sistema que utilize a OO é a identificação das entidades. Uma boa forma para identificá-las é fazer uma análise dos substantivos e/ou frases com substantivos que compõem uma simulação com AGs. As informações capturadas no capítulo anterior (Capítulo 2), bem como as terminologias exibidas no Quadro 1, guiaram este processo de identificação de entidades. As entidades identificadas a partir destes conceitos serão mencionadas daqui por diante como classes de objetos.

Para construir X-GAT, foram identificadas as classes mostradas no diagrama da Figura 4. A classe principal da solução é a superclasse *GeneticAlgorithm*, que concentra informações comuns a todas as implementações de AGs. Esta superclasse possui as subclasses *ClassicGeneticAlgorithm* e *MiuPlusLambdaGeneticAlgorithm*, que são implementações diferentes de AGs. A classe *Population*, que está associada a uma execução de um AG, possui um conjunto de cromossomos que serão administrados durante uma execução de um AG. A classe *Chromossome* contém informações sobre um cromossomo manipulado em uma população, tendo um campo *fitness*, responsável pelo armazenamento da nota de aptidão desta solução. Cada cromossomo possui um conjunto de genes, sendo então criada a classe *Gene*, responsável apenas pelo armazenamento do valor de cada um dos genes da solução. Cada gene pode ser de um tipo, sendo necessária assim a criação das subclasses *IntegerGene*, *RealGene* e *CharGene*, que armazenam o valor bruto de cada gene. Uma simulação com AGs necessita de um método de inicialização (*InitializationMethod*), um de mutação (*MutationMethod*), um de *crossover* (*CrossoverMethod*), um de seleção (*SelectionMethod*) e um de avaliação (*PopulationEvaluationMethod*). Para cada uma

destas necessidades foi criada uma superclasse com um método abstrato de execução. A partir deste método abstrato, cada funcionalidade distinta do algoritmo pode ser implementada como uma subclasse destas superclasses. Por exemplo, quando for necessário a criação de um novo método de seleção basta que a superclasse *SelectionMethod* seja especializada, e assim, o seu método abstrato deverá ser implementado da forma como o usuário queira implementá-la. Esta metodologia torna fácil a adição de novas funcionalidades à ferramenta, sendo estas questões mais bem detalhadas na Seção 3.2.1. Os métodos abstratos das classes que compõem um AG são mostrados na próxima seção, que aborda questões mais específicas de implementação.

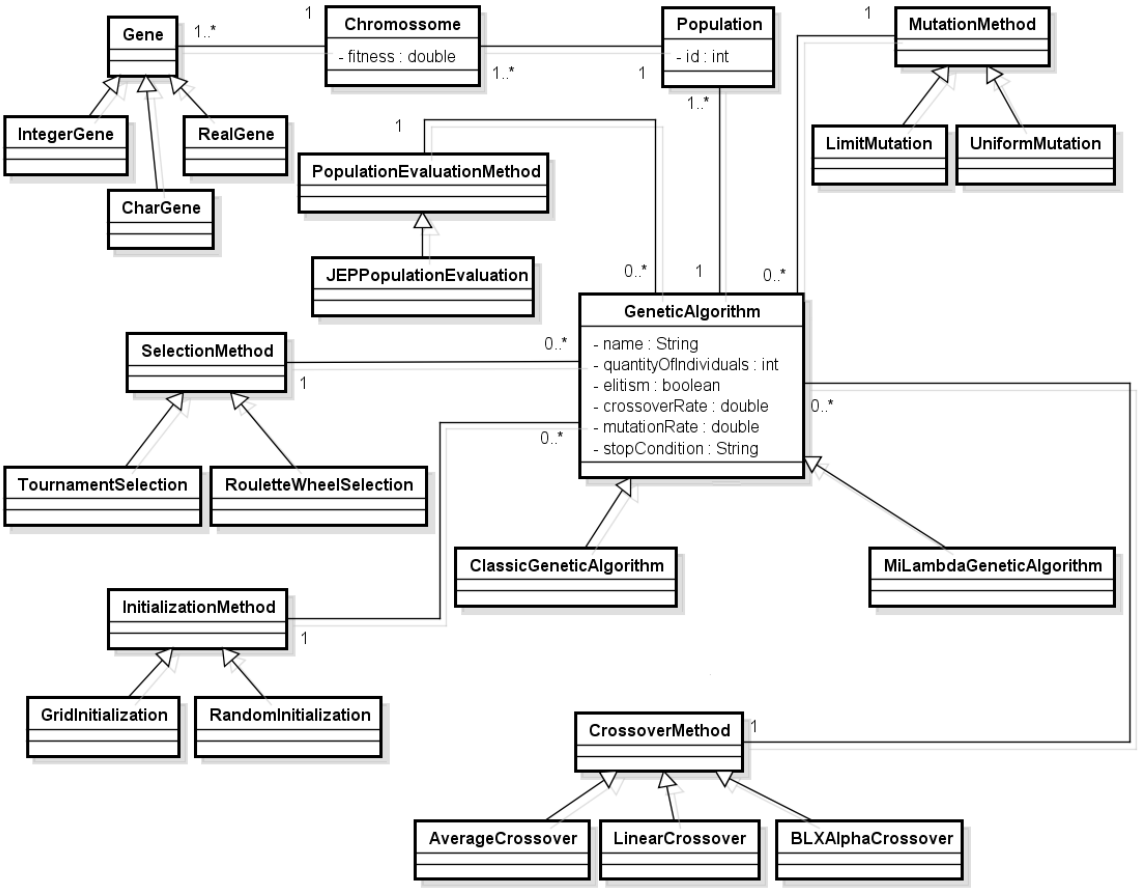


Figura 4 - Diagrama de classes da ferramenta X-GAT.

3.2 Utilização de Padrões de Projeto para Construção do X-GAT

Os padrões de projeto foram criados com o objetivo de retratar e reutilizar as experiências adquiridas por bons projetistas de sistemas, sendo estes projetos construídos com o paradigma OO. Estes padrões resolvem problemas de projeto

específicos e tornam o paradigma OO mais flexível, elegante e reutilizável (Gamma et al., 1999).

De acordo com Gamma et al. (1999), um padrão de projeto possui quatro elementos essenciais:

- **O nome do padrão:** é um identificador que pode ser usado para descrever um problema de projeto, sua solução e conseqüências em uma palavra ou duas;
- **O problema:** descreve quando aplicar o padrão de projeto;
- **A solução:** descreve os elementos que compõem o projeto, suas relações, responsabilidades e colaborações;
- **As conseqüências**: são os resultados e as conseqüências da aplicação do padrão de projeto na solução.

Os padrões de projeto podem ainda ser subdivididos em três tipos:

- **Padrões de projeto criacionais:** descrevem as técnicas para instanciar objetos (ou grupos de objetos);
- **Padrões de projeto estruturais:** permitem que os projetistas organizem classes e objetos em estruturas maiores;

Padrões de projeto comportamentais: atribuem responsabilidades a classes e objetos.

Neste trabalho foram usados dois tipos de padrões de projeto, um padrão de projeto criacional (*Factory Method*) e um padrão de projeto estrutural (*Facade*). As próximas subseções detalham como estes padrões foram utilizados na construção do X-GAT.

3.2.1 O padrão de projeto *Factory Method*

O padrão de projeto *Factory Method* tem por objetivo fornecer uma interface para a criação de um objeto, entretanto, são as subclasses que decidem qual classe será realmente instanciada (Gamma et al., 1999). Assim, este padrão de projeto proporciona uma classe de decisão, a qual retorna uma das muitas possíveis subclasses de uma classe base abstrata, dependendo de um dado que é fornecido como argumento.

No contexto da implementação do X-GAT, este padrão de projeto foi utilizado para implementar as questões de configurabilidade permitidas ao usuário. Existem diversas partes dos AGs que podem ser modificadas de acordo com a necessidade do problema a ser resolvido. Um exemplo são os métodos de *crossover*, pois existem diversas implementações propostas e validadas na literatura. A partir das configurações

definidas pelo usuário da ferramenta X-GAT, que são descritas a partir de um arquivo XML (seção 3.5), o método de *crossover* utilizado na otimização é explicitamente informado neste arquivo. A partir deste arquivo XML, a ferramenta verifica qual é o método de *crossover* escolhido e o aplica na otimização. A forma de descrever as configurações através de um documento XML será mostrada em uma seção 3.5.1. Além do método de *crossover*, outras cinco partes do algoritmo utilizam este padrão de projeto criacional: método de mutação (*MutationMethod*); método de inicialização (*InitializationMethod*); método de seleção (*SelectionMethod*); método de avaliação (*PopulationEvaluationMethod)*. Todas essas partes do algoritmo podem ser combinadas de diversas maneiras, cabendo ao usuário, escolher cada parte e combiná-las da forma mais adequada ao problema de otimização a ser resolvido.

A Figura 5 mostra em notação UML como o *Factory Method* foi utilizado na implementação do X-GAT. Utilizamos o mesmo exemplo anterior, a respeito dos métodos de *crossover*, para mostrar como o padrão de projeto foi aplicado. Ao se aplicar um método de *crossover* em uma execução de uma otimização com um AG, seja utilizando o X-GAT como um framework ou executando um dos AGs prontos para uso, deve-se invocar a classe *CrossoverMethodFactory* passando como argumento uma string com o nome da subclasse a ser instanciada. Como mostrado no diagrama da Figura 5, a classe CrossoverMethod possui um método abstrato (*applyCrossover*) que deve ser implementado por suas subclasses. Este método abstrato recebe dois indivíduos como argumento e retorna um novo indivíduo, gerado a partir destes dois recebidos como argumentos do método.

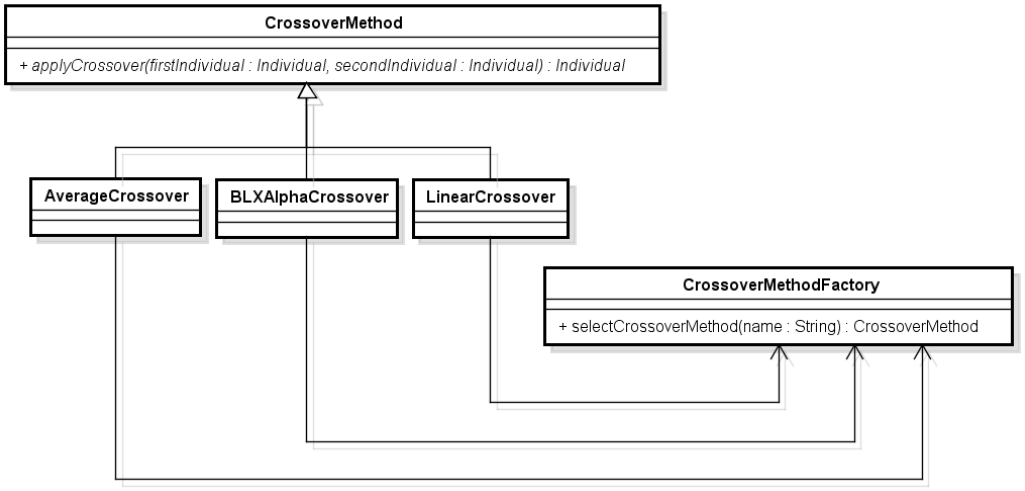


Figura 5 - Utilização do padrão de projeto *Factory Method* na construção da ferramenta X-GAT.

Utilizando o padrão de projeto *Factory Method,* a classe a ser retornada não é importante para o programador da aplicação, existindo assim uma única interface para o método, mas diferentes implementações deste método. Dessa forma, a partir das informações contidas na descrição XML, as partes do AG podem ser selecionadas e aplicadas sem a necessidade do usuário da ferramenta se preocupar em selecionar diretamente no código, a implementação da classe que ele deseja utilizar.

Outro objetivo a ser atingido pela ferramenta é que esta seja extensível. De acordo com tal princípio, novas funcionalidades podem ser adicionadas facilmente à ferramenta X-GAT, sendo este também um princípio que guia a criação de *frameworks*. Para que a extensão das funcionalidades desta ferramenta fosse algo simples a ser realizado, foi utilizada a API de reflexão (http://download.oracle.com/javase/tutorial/reflect/) da linguagem de programação Java. Esta API pode ser utilizada quando é necessário acesso a informações que representam classes e objetos que estão em execução em um programa Java de forma dinâmica. Informações passadas como argumento para a classe de decisão do padrão de projeto *FactoryMethod* podem ser utilizadas para fazer a instanciação da subclasse a ser usada na otimização. Por exemplo, caso seja necessária a criação de um novo método de *crossover*, o usuário não precisa ir diretamente ao código da classe *CrossoverMethodFactory* para adicionar esta nova subclasse. Na descrição XML da otimização a ser realizada, basta o usuário informar qual é exatamente a subclasse da classe CrossoverMethod que deve ser instanciada. A subclasse informada na descrição é instanciada de forma dinâmica, podendo assim ser utilizada em qualquer otimização realizada pela ferramenta X-GAT.

3.2.2 Os Pacotes da Ferramenta X-GAT

Em UML, um pacote é um mecanismo de propósito geral para organizar elementos de modelagem em grupos (Booch *et al., 1999*). Esta representação será adotada para facilitar o entendimento do padrão de projeto *Facade*, apresentado na próxima seção. Com o objetivo de facilitar este entendimento, as classes criadas para a ferramenta X-GAT são inseridas em pacotes, e são agrupadas de acordo com semelhanças existentes entre elas. Dessa forma, foram criados os seguintes pacotes:

- **Algorithms:** este pacote contém classes que executam uma otimização com AGs;
- **GA Types:** este pacote contém classes que são manipuladas durante uma execução de uma otimização com AGs;

- **Evaluation:** este pacote contém classes que calculam funções de aptidão para a execução de uma otimização com AGs;
- **Evolution:** este pacote contém classes com operadores de reprodução (como *crossover* e mutação) e classes com operadores seleção;
- **Initialization:** este pacote contém classes que inicializam a primeira população de indivíduos em uma otimização com AGs;
- **Factories:** este pacote contém as classes com as fábricas criadas pelo padrão de projeto *Factory Method*.

Com o objetivo de representar as relações de dependência entre estes pacotes, foi utilizado o diagrama de pacotes da UML. Os pacotes criados para a ferramenta X-GAT possuem a relação de dependência mostrada na Figura 6. O pacote *GA Types* possui as classes que armazenam as estruturas de dados manipuladas na execução de um AG. Os pacotes *Evaluation*, *Evolution* e *Initialization* manipulam as classes inseridas no pacote *GA Types* para implementar o núcleo do código da heurística de busca. Já o pacote *Factories* importa os pacotes *Evaluation, Evolution* e *Initialization*, pois este pacote contém as classes responsáveis pela inserção do padrão de projeto *Factory Method*. Cada uma das classes do pacote *Factories* deve selecionar qual das subclasses deve ser instanciada na simulação que deverá ser executada. Por exemplo, o usuário tem a opção de escolher vários métodos de mutação. A partir da operação *selectMutationMethod*, uma subclasse derivada da classe *MutationMethod* será instanciada para a execução da heurística de busca. Por último, o pacote *Algorithms* tem dependência do pacote *Factories,* pois é o pacote *Algorithms* que contém as classes principais que executam uma busca pelo ótimo global com AGs, necessitando assim invocar as partes configuráveis do algoritmo e também gerenciar a seqüência de execução das operações da heurística de busca. A seção seguinte mostra o padrão de projeto utilizado para gerenciar essa seqüência de execução de operações.

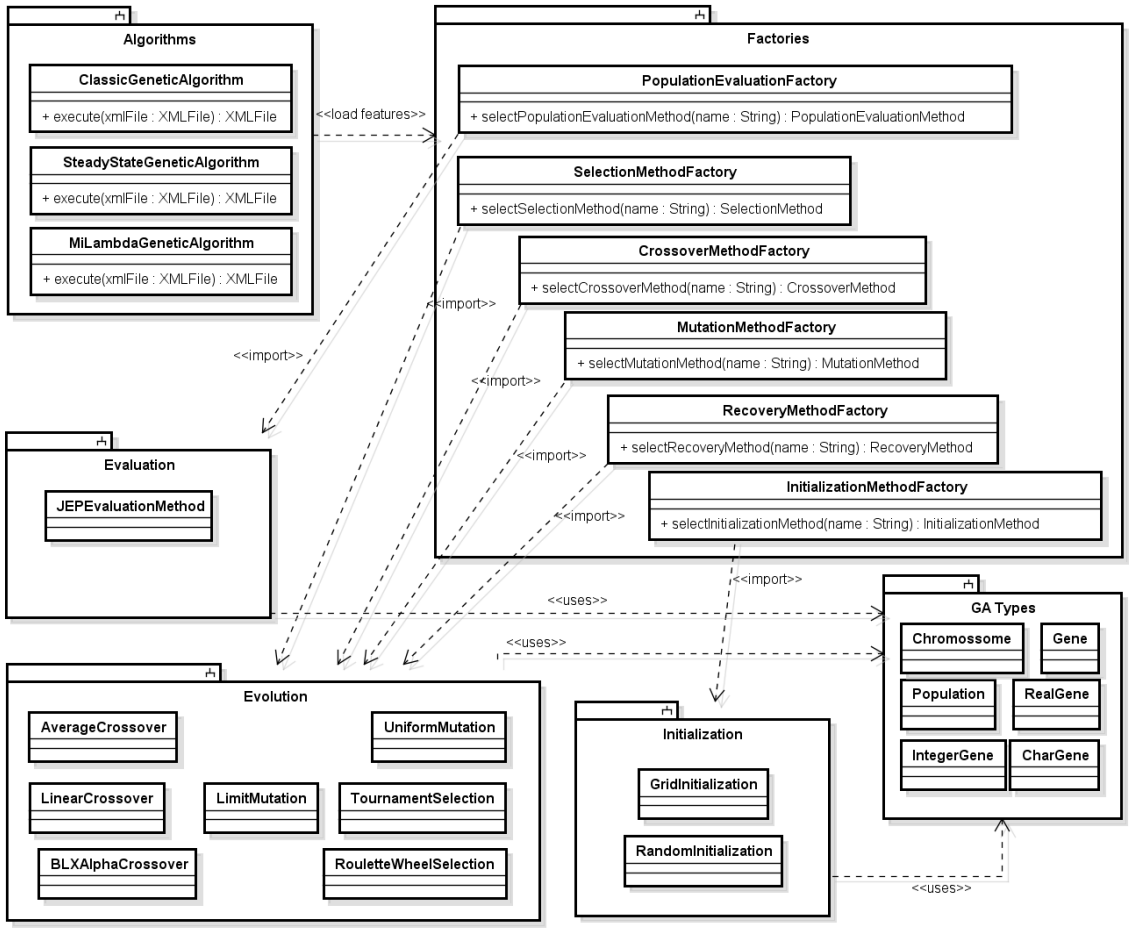


Figura 6 - Diagrama de pacotes da ferramenta X-GAT.

3.2.3 O padrão de projeto *Facade*

O padrão de projeto *Facade* fornece uma interface unificada para um conjunto de interfaces em um subsistema (Gamma et al., 1999). Dessa forma, este padrão de projeto define uma interface de nível mais alto que torna o subsistema mais fácil de usar.

Na ferramenta X-GAT, o padrão de projeto *Facade* foi utilizado para realizar o processo de otimização com AGs a partir de uma descrição XML criada pelo usuário. A partir dessa descrição, as funcionalidades são carregadas e utilizadas com o objetivo de realizar a busca pelo ótimo global. Ao se introduzir este padrão de projeto, a seqüência de operações necessárias a execução do AG fica transparente, sendo necessária apenas a invocação dos métodos expostos pela interface criada pelo padrão de projeto.

Um dos objetivos da ferramenta X-GAT também é fornecer um framework para otimização com AGs. Dessa forma, foram também expostas outras operações na camada criada pelo padrão de projeto *Facade*. Além de executar uma otimização com AGs, a

partir de uma descrição XML, as funcionalidades do X-GAT podem ser usadas para criar outras metodologias de otimização com AGs. As operações disponíveis para uso na ferramenta podem ser vistas na Figura 7 e são explicadas a seguir. Quando o usuário quer realizar uma otimização, ele deve primeiramente descrever todas as informações necessárias à execução deste processo em um arquivo a XML. Escrito este arquivo, ele poderá invocar a operação *executeGeneticAlgorithm* que recebe um arquivo XML como argumento. Esta função irá utilizar todas as informações contidas neste arquivo XML e realizará a otimização. Quando a ferramenta X-GAT for utilizada como um *framework*, as operações *loadInitializationMethod*, *loadEvaluationMethod*, *loadCrossoverMethod*, *loadMutationMethod* e *loadSelectionMethod* podem ser utilizadas. Essas cinco operações invocam partes específicas do fluxo de execução de um AG, podendo ser reusadas na construção de novas heurísticas de otimização com AEs ou AGs.

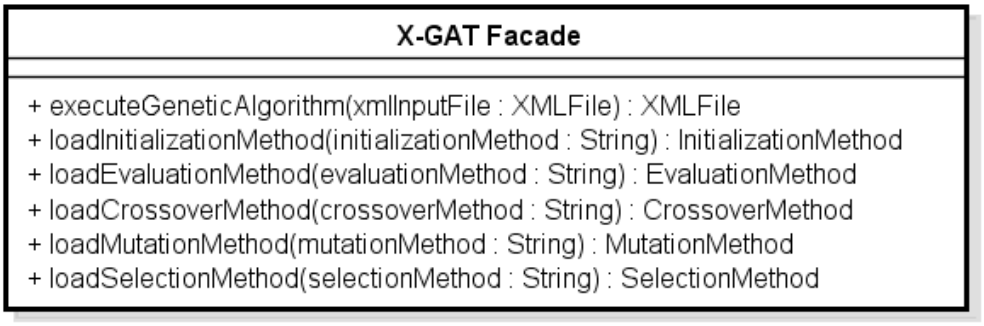


Figura 7 - Operações disponíveis na classe X-GAT Facade.

A Figura 8 mostra o impacto da inserção do padrão de projeto *Facade* na ferramenta X-GAT. A partir da interface unificada prevista pelo padrão de projeto *Facade*, qualquer processo de otimização pode ser realizado pela operação *executeGeneticAlgorithm*. Esta operação recebe um arquivo XML como entrada que contém a especificação das funcionalidades e dados necessários para que o processo de otimização seja executado. Como saída deste método, é gerado outro arquivo XML, contendo informações sobre o que foi encontrado como a melhor solução para o problema e informações sobre as gerações evoluídas durante todo o processo de otimização. Já as outras operações devem ser utilizadas quando o X-GAT for utilizado como um framework. Essas operações simplesmente carregam a subclasse adequada à requisição feita pela string passada como argumento.

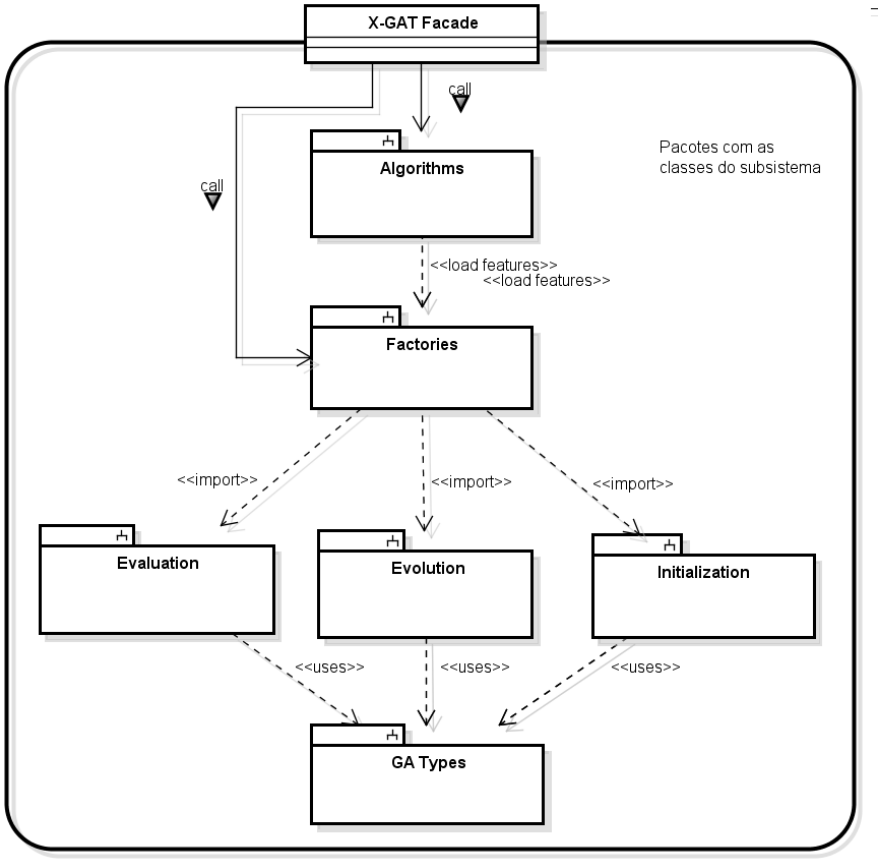


Figura 8 - Utilização do padrão de projeto *Facade* na construção da ferramenta X-GAT.

Com o objetivo de ilustrar a execução de uma otimização com AGs, é mostrado na Figura 9 um diagrama de colaboração de como uma execução do algoritmo genético clássico é realizada. Um diagrama de colaboração exibe uma interação, consistindo de um conjunto de objetos e seus relacionamentos (Booch *et al*, 1999). Através da operação *executeGeneticAlgorithm* exposta pelo padrão de projeto *Facade*, a descrição com os dados da otimização é passada para a classe que carrega o AG a ser executado. Neste exemplo, é utilizado como exemplo o AG clássico, que segue o fluxo exibido em detalhes pela Figura 1. Após a escolha do AG descrito no arquivo XML, é invocado o método *executeGeneticAlgorithm* que é responsável pela execução da heurística. Primeiramente, são selecionadas as funcionalidades a serem utilizadas no processo de otimização. Para que as classes das funcionalidades sejam carregadas adequadamente, é solicitado a cada classe criada a partir do padrão de projeto *FactoryMethod* (*InitializationMethodFactory, SelectionMethodFactory, MutationMethodFactory, PopulationEvaluationFactory, RecoveryMethodFactory e CrossoverMethodFactory*) a

subclasse solicitada no arquivo de descrição XML. A partir de cada subclasse instanciada de forma correta, a execução do AG pode ser realizada de forma adequada. Após a execução da heurística, os resultados da otimização podem ser vistos em um arquivo XML de saída, que contém o resultado completo da busca realizada. Os detalhes dos arquivos XML de entrada e saída serão vistos na Seção 3.5.

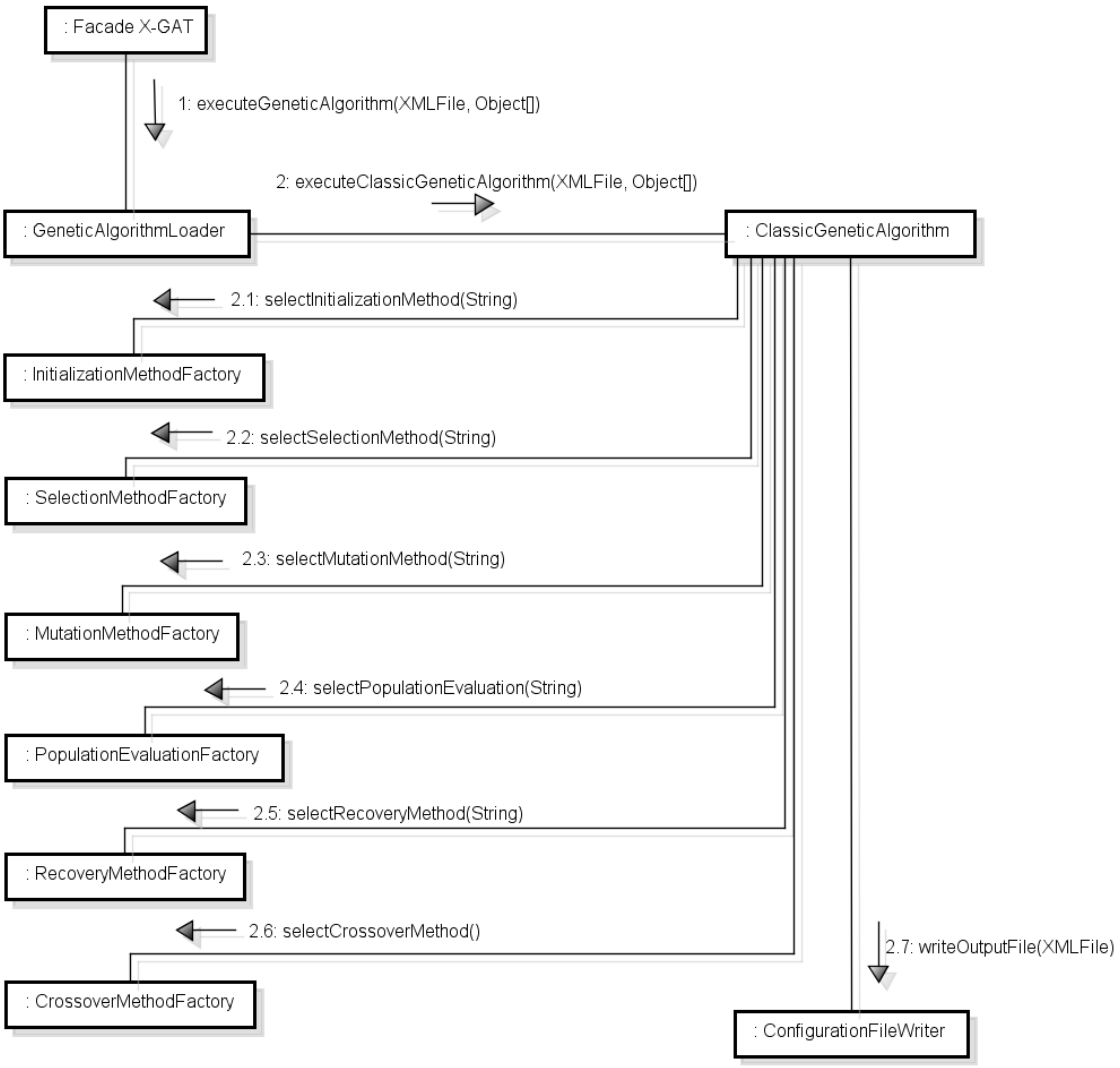


Figura 9 - Diagrama de Colaboração para o processo de execução de um AG clássico.

3.3 Os Algoritmos Genéticos Implementados

A ferramenta X-GAT se diferencia das outras ferramentas disponíveis (Seção 2.3.2) porque oferece alguns AGs implementados e prontos para uso. Esta seção descreve as similaridades e as diferenças entre os dois AGs implementados para a ferramenta X-

GAT. Os dois AGs implementados podem utilizar as funcionalidades descritas na seção 3.4. A diferença entre esses dois AGs está na forma com que as populações são gerenciadas durante a execução de uma otimização, ou seja, nos critérios utilizados para que os indivíduos de uma geração sobrevivam para compor a geração posterior.

3.3.1 O Algoritmo Genético Clássico

O AG clássico implementado pela ferramenta X-GAT é baseado nos primeiros trabalhos propostos por Holland (1975). Este AG segue os mesmos princípios explicados na Seção 2.2.1. Dada certa geração (*t*), todos os indivíduos desta geração (*t*) são substituídos por novos indivíduos na próxima geração (*t*+1). Assim, boas soluções (indivíduos) geradas na geração *t* podem deixar de existir na geração (*t*+1), havendo a possibilidade destas soluções não aparecerem mais durante o processo de otimização. Essa perda de soluções não reproduz fielmente o que acontece na natureza, pois indivíduos de uma nova geração podem procriar com indivíduos de gerações anteriores. Entretanto, este tipo de solução faz com que não ocorra convergência primária de soluções, problema levantado na Seção 2.2.3.

Esta metodologia não é indicada quando o tempo de resposta para a otimização deve ser curto, pois o problema convergência primária é tratado nesta metodologia através da geração de um conjunto de novos indivíduos, gerados a partir da população anterior. Quando se necessita de um tempo de resposta mais curto, o outro AG implementado (Algoritmo Genético (μ + λ)) pela ferramenta X-GAT pode ser usado.

3.3.2 O Algoritmo Genético (μ + λ)

Não existe impedimento para que a próxima geração seja selecionada a partir dos melhores indivíduos da geração anterior. Utilizando esta afirmativa, a abordagem (μ + λ) faz com que existam sempre μ indivíduos da população de certa geração (*t*) na geração (*t*+1). O valor de λ representa a quantidade de novos indivíduos por geração. O conjunto de todos estes indivíduos (μ + λ indivíduos) competem entre si de forma que apenas os μ melhores indivíduos de melhor valor de nota de aptidão sobrevivem para a próxima geração. Dessa forma, serão acrescentados os μ melhores indivíduos da geração (*t*) na geração (*t*+1), onde geralmente μ < λ.

Essa metodologia favorece uma convergência rápida dos indivíduos para certa região do espaço de busca, havendo assim perda de diversidade nas soluções do problema. Além disso, estarão sendo sempre administrados μ indivíduos a mais durante

a reprodução em cada iteração do AG, aumentando assim a quantidade de memória necessária para a execução da otimização.

3.4 Funcionalidades do X-GAT

A ferramenta X-GAT se propõe a implementar diversas funcionalidades presentes na literatura dos AGs. As funcionalidades nada mais são do que as subclasses que implementam cada parte do AG, gerenciadas pelas classes do padrão de projeto *Factory Method*. Esta seção tem o objetivo de descrever em detalhes cada uma destas funcionalidades disponíveis na ferramenta de otimização X-GAT.

3.4.1 Métodos de Inicialização

Um método de inicialização em um AG é a forma pela qual a primeira população de indivíduos será criada no processo de otimização. Foram implementadas duas formas de inicialização: a inicialização randômica (aleatória) e a inicialização por *grid*. Estas duas formas de inicializar o AG foram propostas desde os trabalhos pioneiros na área (Holland, 1975; De Jong, 1975), não existindo referências seminais que sejam de nosso conhecimento aos autores desta metodologia na literatura sobre AGs.

A inicialização randômica é a forma mais simples de inicializar um processo de otimização com AGs. Cada indivíduo é gerado de forma aleatória, não existindo um critério para que a primeira população seja criada. Dada certa quantidade n de indivíduos que irão competir em um ambiente, serão gerados esses n indivíduos no espaço de busca do problema. Assim, áreas do espaço de busca podem não ser exploradas no início do AG, ficando condicionadas à exploração apenas se os métodos de reprodução conseguirem encontrar estas áreas do espaço de busca, não havendo assim garantia de que serão exploradas durante o processo de otimização.

Já a inicialização por *grid* segue o princípio de que na primeira população serão gerados indivíduos igualmente espaçados no espaço de busca. Isso significa dizer que o espaço de busca é explorado uniformemente na primeira população de indivíduos, não havendo assim regiões inexploradas, evitando que haja uma convergência muito rápida do processo de otimização para certa região do espaço de busca.

A Figura 10 mostra as duas metodologias de inicialização para uma otimização com uma quantidade de 25 indivíduos por geração. Utilizaremos como exemplo um problema qualquer que possui duas variáveis de entrada a serem otimizadas. O espaço de busca para as duas variáveis está condicionado ao limite inferior de valor 0, e com

limite superior de valor 2. Na Figura 10 (a) é mostrada a inicialização randômica dos valores. Como pode ser visto, não existe um critério para gerar as soluções (indivíduos) do problema. Já a Figura 10 (b) mostra a inicialização por *grid*, na qual as soluções são espalhadas no espaço de busca de modo uniforme.

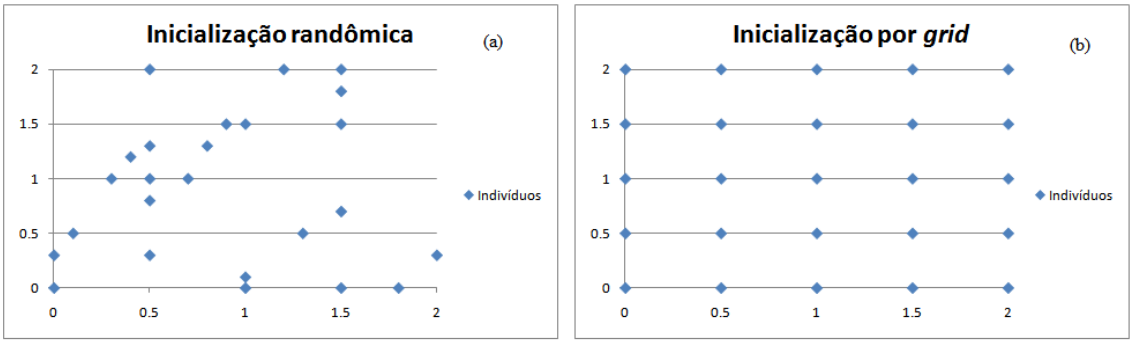


Figura 10 – (a) Método de inicialização randômica. (b) Método de inicialização por *grid*.

3.4.2 Método de Seleção

Um método de seleção é o processo pelo qual os indivíduos mais aptos serão escolhidos para realizar o processo de reprodução. Estes métodos utilizam como valor de comparação a nota de aptidão atribuída a cada indivíduo em uma geração. Na ferramenta X-GAT, duas metodologias de seleção foram implementadas: a seleção por torneio e a seleção por roleta viciada. Assim como os métodos de inicialização, estas duas metodologias de seleção foram propostas desde os primeiros trabalhos com AGs, não existindo também referências seminais que sejam de nosso conhecimento sobre os autores na literatura de AGs.

Na seleção por torneio, são escolhidos n indivíduos da população aleatoriamente. O valor de n deve ser sempre menor do que o tamanho total da população que está sendo trabalhada. Após a seleção destes n indivíduos, é realizado um torneio entre eles e o indivíduo selecionado será aquele que possuir a melhor nota de aptidão.

A Figura 11 mostra um exemplo de uma seleção por torneio com $n = 3$. Os indivíduos B, D e E são selecionados aleatoriamente e competem entre si em um torneio. Se o processo de otimização fosse de minimização, o indivíduo escolhido para que os operadores de reprodução fossem aplicados seria o indivíduo B. Já se o problema fosse de maximização, o indivíduo selecionado seria o E.

Indivíduo	Nota de aptidão
A	3.5
B	5
C	7
D	10
E	13.4
F	14.5

Figura 11 - Seleção por Torneio.

Já na seleção por roleta viciada, cada indivíduo recebe uma porcentagem de probabilidade de ser selecionado de forma proporcional a sua nota de aptidão. Isso significa que os indivíduos que possuem uma melhor nota de aptidão terão uma maior probabilidade de serem escolhidos para gerar descendentes. A partir dos valores de aptidão, é construída uma roleta (virtual) na qual cada indivíduo recebe um pedaço proporcional a sua nota de aptidão, onde estes valores somados não devem exceder 100%.

A Figura 12 mostra um exemplo da metodologia de seleção por roleta viciada. Cada um dos 6 indivíduos recebe uma nota proporcional a sua nota de aptidão. Dados os valores da porcentagem da roleta virtual a cada indivíduo, um número é sorteado aleatoriamente e o indivíduo selecionado para a reprodução será aquele que possuir a nota de aptidão maior ou igual àquele valor. Por exemplo, se o número sorteado foi 19.7, o indivíduo selecionado para reprodução seria o indivíduo E.

Indivíduo	Nota de aptidão	Parte da roleta (%)
A	3.5	6.6%
B	5	9.4%
C	7	13.2%
D	10	18.7%
E	13.4	25%
F	14.5	27.1%
Total	53.4	100%

Figura 12 – Seleção por roleta viciada.

3.4.3 Métodos de Crossover

Um método de *crossover* deve tentar reproduzir de alguma maneira o processo de troca de informações genéticas entre dois indivíduos. Este processo pode ser simulado computacionalmente pela troca de bits entre dois indivíduos, explicada detalhadamente na Seção 2.2.1. Entretanto, codificar a solução de problemas por meio de números binários é custoso, lento e os resultados não são visíveis de forma clara e objetiva. Para evitar estes problemas, na ferramenta X-GAT foi utilizado o processo de codificação com números de ponto flutuante, amplamente utilizado e difundido por trabalhos de Janikow & Michalewicz (1991), & Bramlette & Cusic (1989), Bramlette (1991) e Furuya & Haftka (1993).

Na ferramenta X-GAT, foram implementados três métodos de *crossover* para ponto flutuante: o *crossover* média (Davis, 1991), o *crossover* BLXAlpha (Eshelman & Shaffer, 1993) e o *crossover* Linear (Wright, 1991).

O *crossover* média (Davis, 1991), assim como o nome sugere, é gerado a partir da média aritmética entre os dois valores dos parâmetros do indivíduo. Este processo tende a levar os genes para o centro do espaço de busca, causando perda de diversidade. Entretanto, esta metodologia não extrapola os limites para além da região da população inicial, fazendo com que o descendente desse pai não seja descartado por estar fora da região limitada pelo espaço de busca.

A Figura 13 mostra um exemplo de utilização do *crossover* média. Assim como no exemplo da Figura 10, utilizaremos um problema qualquer que possui duas variáveis de entrada a serem otimizadas e também o mesmo espaço de busca. Este exemplo é utilizado também para ilustrar o funcionamento dos outros dois tipos de *crossover*. Como pode ser visto na Figura 13, os pais escolhidos para a reprodução são os pares ordenados p1(0.5, 0.5) e p2 (1.5, 1.5). O indivíduo gerado pelo *crossover* média é aquele cujo par ordenado é a média aritmética dos pais, ou seja, o ponto (1.0, 1.0). Para realizar este procedimento, utilizamos a Equação 1.

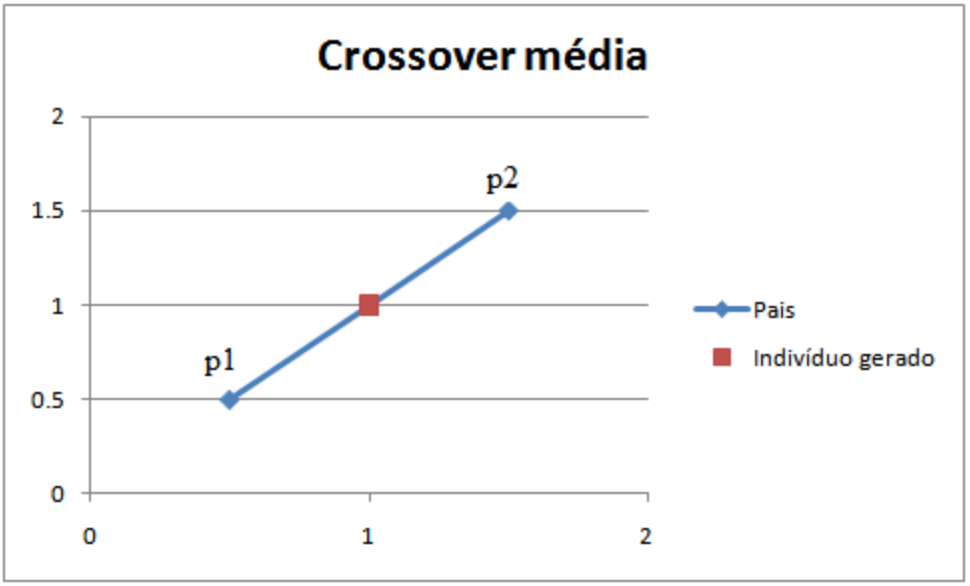


Figura 13 - *Crossover* média.

Dados dois indivíduos i1 e i2:

i1 = (p11, p12,..., p1n)

i2 = (p21, p22,..., p2 n)

é produzido um cromossomo tal que:

$$\mathbf{c}_i = (p1_i + p2_i)/2$$

onde i = 1, 2,..., n; e **c** = (c1, c2,..., cn).

Equação 1 – Equação do *crossover* média.

O crossover BLX-Alpha (Eshelman & Shaffer, 1993), diferentemente do *crossover* média, não tende a levar os genes para o centro do espaço de busca. Nessa metodologia, os novos cromossomos são gerados com certa aleatoriedade, simulando basicamente o que ocorre nos *crossovers* com base binária. No *crossover* BLX-Alpha, os novos indivíduos são gerados de acordo com a Equação 2. O valor de β é sorteado toda vez que um novo descendente é gerado, criando assim uma maior variabilidade genética nos descendentes obtidos por meio desta abordagem.

Dados dois indivíduos i_1 e i_2, é produzido um cromossomo tal que:

$$c = i_1 + \beta (i_2 - i_1)$$

onde: $\beta \in U(-\alpha, 1 + \alpha)$

com α igual a 0.5 e

U representa uma distribuição uniforme.

Equação 2 - Equação do crossover BLX-Alpha.

A Figura 14 mostra um exemplo do que pode ser gerado pelo *crossover* BLX-Alpha. Ao ser calculado aleatoriamente o valor para β dentro dos limites definidos por α, o novo indivíduo pode ser gerado. Note-se que a introdução do valor de α aumenta a quantidade de valores que podem ser gerados para o descendente do procedimento. Ainda, o fato do valor de β ser obtido cada vez que o método é aplicado, aumenta de forma considerável, a variabilidade genética das soluções, pois pais iguais podem gerar filhos diferentes, desde que o valor sorteado para β seja diferente durante a execução deste procedimento.

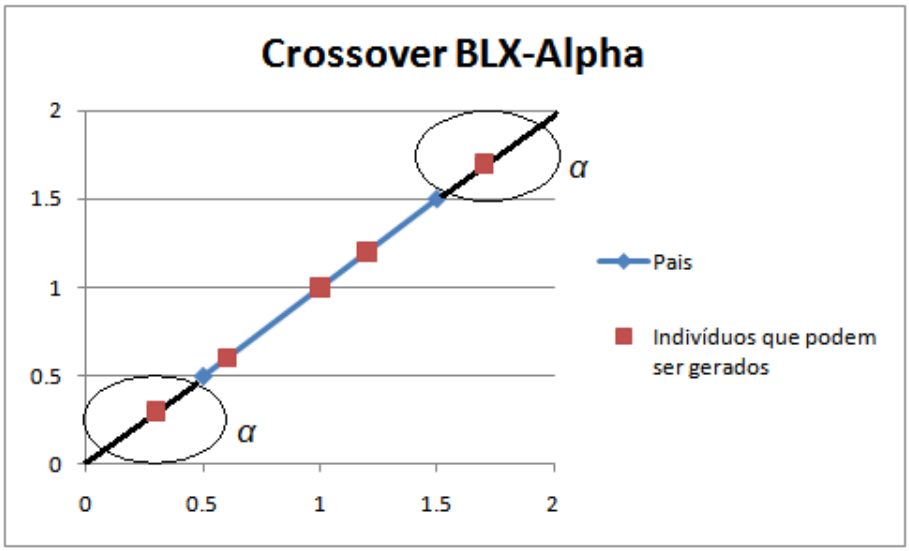


Figura 14 - Crossover BLX-Alpha.

Já o *crossover* Linear (Wright, 1991), diferente dos outros dois tipos de *crossovers* já mencionados, insere um cálculo prévio da função de aptidão de cada possível indivíduo a ser gerado. Este *crossover* gera três filhos de acordo com a Equação 3.

Dados dois indivíduos i_1 e i_2, são produzidos três cromossomos (c_1, c_2 e c_3) tais que:

$$c_1 = (0.5)*i_1 + (0.5) * i_2$$

$$c_2 = (1.5)*i_1 - (0.5) *i_2$$

$$c_3 = (-0.5)*i_1 + (1.5)*i_2$$

onde, apenas **o cromossomo** que possui **a melhor nota de aptidão** é **retornado** como resultado.

Equação 3 – Equações do *crossover* Linear.

A Figura 15 mostra um exemplo dos indivíduos gerados pelo método de *crossover* Linear. Dados os dois pais para aplicação da metodologia, serão gerados três filhos de acordo com as equações mostradas. Como pode ser visto na Figura 15, os filhos são gerados deterministicamente e apenas o melhor dos três é retornado como resultado, ou seja, o que tiver o melhor valor de função de aptidão é retornado como resultado da execução do método de *crossover*. Nota-se que esta metodologia sempre produz filhos iguais para os mesmos pais, perdendo-se então variabilidade genética nas soluções geradas. Entretanto, o fato de explorar qual dos três filhos produz um melhor resultado faz com que as soluções possuam uma boa probabilidade de convergir para o lugar certo, diferente do *crossover* média que não explora esta questão e do BLX-Alpha, que não verifica a nota aptidão do indivíduo gerado.

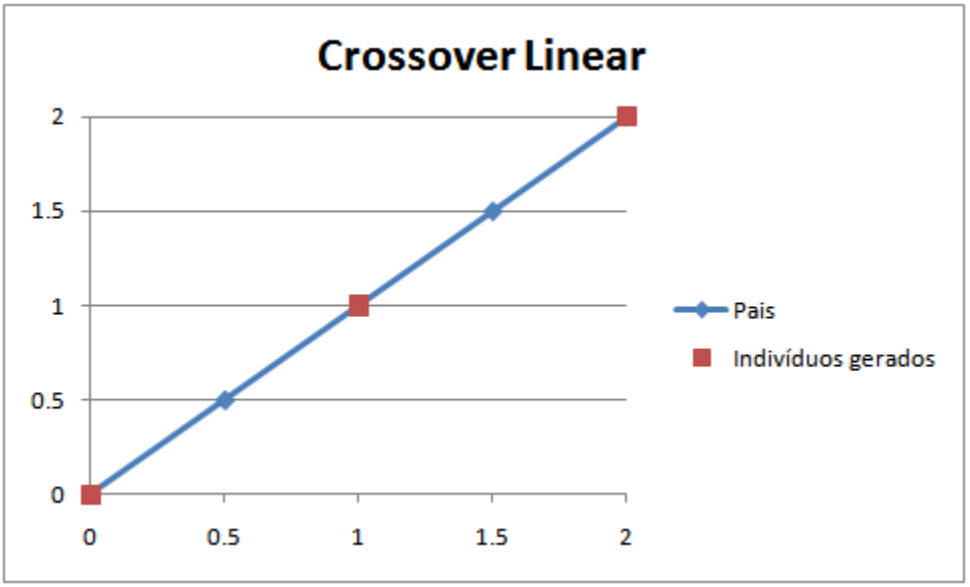


Figura 15 - *Crossover* Linear.

3.4.4 Métodos de Mutação

Uma operação de mutação tem por objetivo evitar que as soluções encontradas pela execução do AG fiquem presas em mínimos ou máximos locais. A mutação é efetuada alterando-se o valor de um gene de um indivíduo. Cada gene deste indivíduo possui uma probabilidade de ser alterado (taxa de ocorrência de mutação). Dessa forma, vários indivíduos da nova população gerada podem ter um de seus genes alterados aleatoriamente. Foram implementados na ferramenta X-GAT dois métodos de mutação: a mutação limite (Michalewicz, 1999) e a mutação uniforme (Michalewicz, 1999).

Na mutação limite (Michalewicz, 1999), o gene que deve sofrer mutação é substituído por um dos limites do intervalo factível (limite inferior da variável de entrada ou limite superior da variável de entrada). Os possíveis valores gerados pela metodologia de mutação uniforme são mostrados na Equação 4. Dessa forma, são gerados apenas dois valores neste procedimento. Caso o valor sorteado *r* seja um valor menor que 0.5, o limite inferior do gene que sofre mutação é retornado como resultado do procedimento. Já se o valor de *r* for maior ou igual a 0.5, o limite superior do gene que sofre mutação é retornado como resultado do procedimento.

Equação 4 - Equação da mutação uniforme.

Dados os limites do intervalo factível [x_i, y_i] e dada uma distribuição de probabilidades uniforme U(0,1), e um valor r $\in$ U(0,1):

$$c_i = \begin{cases} x_i, \text{ se } r < 0.5 \\ y_i, \text{ se } r >= 0.5 \end{cases}$$

Já a mutação uniforme (Michalewicz, 1999) substitui o gene que sofre mutação por um valor aleatório no intervalo factível deste gene. Diferente da metodologia anterior, que gera apenas os valores extremos do gene (uma das variáveis de entrada), esta metodologia pode gerar qualquer valor dentro deste limite, aumentando assim a possibilidade de exploração de áreas que ainda não foram investigadas pelo AG.

3.4.5 Métodos de Avaliação

Um método de avaliação é responsável por atribuir a nota de aptidão a um indivíduo em uma população. Problemas de otimização distintos possuem formas de avaliação distintas. Assim, torna-se praticamente impossível criar uma classe genérica para avaliação de indivíduos aplicável em qualquer problema de otimização. Diante

disso, foi criada, baseada no padrão de projeto *Factory Method*, uma classe abstrata (a classe *PopulationEvaluationMethod*, vide Figura 4, que necessariamente deve ser estendida para implementação de técnicas de avaliação específicas. As técnicas de avaliação são implementadas mediante a redefinição de métodos abstratos herdados de subclasses dessa classe abstrata. Assim, a forma adequada de avaliar um indivíduo em um determinado problema de otimização pode ser facilmente definida. Uma grande vantagem da utilização dessa abordagem é que a partir da criação de uma classe de avaliação, fica permitido qualquer tipo de chamada externa ao código do AG, tornando assim viável que outros programas possam ser executados com o objetivo de auxiliar no cálculo da função de aptidão do indivíduo.

Para testar algumas das funcionalidades implementadas pela ferramenta X-GAT e também permitir que funções matemáticas pudessem ser usadas nos testes, mas sem a necessidade de novas classes de avaliação para funções matemáticas distintas, foi criada a classe *JEPPopulationEvaluation*. Essa classe permite que seja passada a expressão da função matemática a ser otimizada no problema como um dado no arquivo XML de entrada, evitando assim a necessidade de alteração de código na classe de avaliação.

O JEP (Java Expression Parser), uma biblioteca Java desenvolvida pela Singular Systems (*http://www.singularsys.com/jep/*), traduz as expressões das funções matemáticas de avaliação de aptidão do arquivo XML de entrada para código Java, que são lidas como linhas de texto. A partir da sintaxe definida pelo JEP (que pode ser vista no Anexo A), o usuário pode descrever a função a ser otimizada no arquivo XML de entrada. Essas linhas são lidas e traduzidas para código Java, permitindo assim que a função matemática seja calculada dinamicamente. Outra vantagem da utilização do JEP é que esta biblioteca possui uma grande quantidade de constantes e funções disponíveis como seno, cosseno, dentre outras.

3.4.6 Critérios de Parada de um AG

Durante o processo de otimização com AGs, é necessário determinar um critério de parada do algoritmo. Vários critérios são utilizados para encerrar um AG, dentre eles, os mais conhecidos são implementados pela ferramenta X-GAT: número finito de gerações evoluídas; convergência dos genes; valor exato do ótimo global; um valor aproximado de algum outro valor.

O critério de parada baseado na quantidade finita de gerações evoluídas é, sem dúvida, o critério mais simples que pode ser adotado em uma otimização com AGs. Nesta

metodologia, dado um valor *x* de gerações a serem evoluídas, o algoritmo irá ser executado *x* vezes, e o melhor resultado da otimização será o melhor indivíduo encontrado na geração (*x*).

Outra forma de parar um AG é verificar a variabilidade genética dos indivíduos da última população evoluída. A variabilidade genética em uma população é uma proporção calculada a partir da diferença entre os valores dos genes desta população. Caso esta variabilidade genética esteja muito pequena, isto significa que os indivíduos desta população estão concentrados em certa região do espaço de busca, onde provavelmente o ótimo local deve estar, podendo este valor ser o ótimo global. Durante a execução da heurística, essa é a área onde os melhores valores para nota de aptidão foram obtidos.

Parar um AG apenas quando o ótimo global é encontrado é algo que a princípio não faz sentido, visto que os AGs devem ser usados apenas quando um problema não possui uma técnica em que este valor ótimo pode ser encontrado. Entretanto, com o objetivo de testar se um AG foi implementado corretamente, esta funcionalidade é bastante útil. Testar AGs em problemas que possuem uma solução conhecida é uma boa prática para validar uma sua implementação, pois se a implementação foi capaz de encontrar este ótimo global, existe então a possibilidade que ela consiga se aproximar bastante ou até mesmo encontrar o ótimo global de um problema desconhecido. Essa funcionalidade foi utilizada nos testes da ferramenta X-GAT (Capítulo 4), e é uma abordagem comum para validação de implementações de AGs, como pode ser visto nos trabalhos de (Santos *et*. al, 2003; Madeiro, 2010; Christiani 2009; Hrstka & Kucerová 2004). Para parar a execução da otimização, esta abordagem que ocorre insere um problema quando o AG fica preso em uma região na qual o valor ótimo não poderá ser encontrado, ficando dependente da ocorrência de uma mutação que retire os indivíduos desta região do espaço de busca.

O critério de parada que avalia se um valor está perto de outro valor é bastante utilizado em problemas de otimização que utilizam funções para minimizar um valor (e. g., minimizar o erro) ou maximizar algo (e. g., maximizar lucros). Suponha um problema cujo cálculo de uma função de erro é utilizado para se computar a nota de aptidão de um indivíduo. Para este problema, desejamos obter uma taxa de erro de pelo menos 15%. Durante a execução do processo de otimização, o AG só irá parar quando o valor de um indivíduo encontrado for de pelo menos 15%, caso contrário, o AG continuará seu processo de busca por um ótimo global. Da mesma forma que o critério anterior, o AG

pode ficar em execução indefinidamente, pois pode acontecer convergência para uma região em que não se obtém esse valor mínimo de 15%. Essa metodologia é ainda mais problemática que a anterior, pois como o valor do ótimo global é desconhecido, o AG pode continuar sendo executado indefinidamente.

3.5 Descrição das funcionalidades do X-GAT com XML

As funcionalidades disponíveis na ferramenta X-GAT para uso em uma otimização com AGs devem estar contidas em um arquivo XML. O principal objetivo da utilização da linguagem de descrição de dados XML é tornar viável a integração do X-GAT com aplicações que sejam implementados em outras linguagens de programação (e. g., C, C++, Fortran) além da linguagem utilizada para implementar a ferramenta X-GAT, que foi a linguagem Java. Este processo é simples, como pode ser visto na Figura 16. Uma otimização pode ser realizada a partir da manipulação dos dados contidos nos arquivos XML de entrada e saída. O problema a ser otimizado (que pode estar implementado em qualquer linguagem de programação) manipula (descreve os dados a serem otimizados e as funcionalidades a serem utilizadas no processo de otimização com o X-GAT) os dados de entrada para uma otimização com AGs. A ferramenta X-GAT carrega essa descrição e executa o processo de otimização. A saída deste processo de otimização é outro arquivo XML, contendo o melhor resultado encontrado e uma série de dados sobre as gerações evoluídas pelo AG.

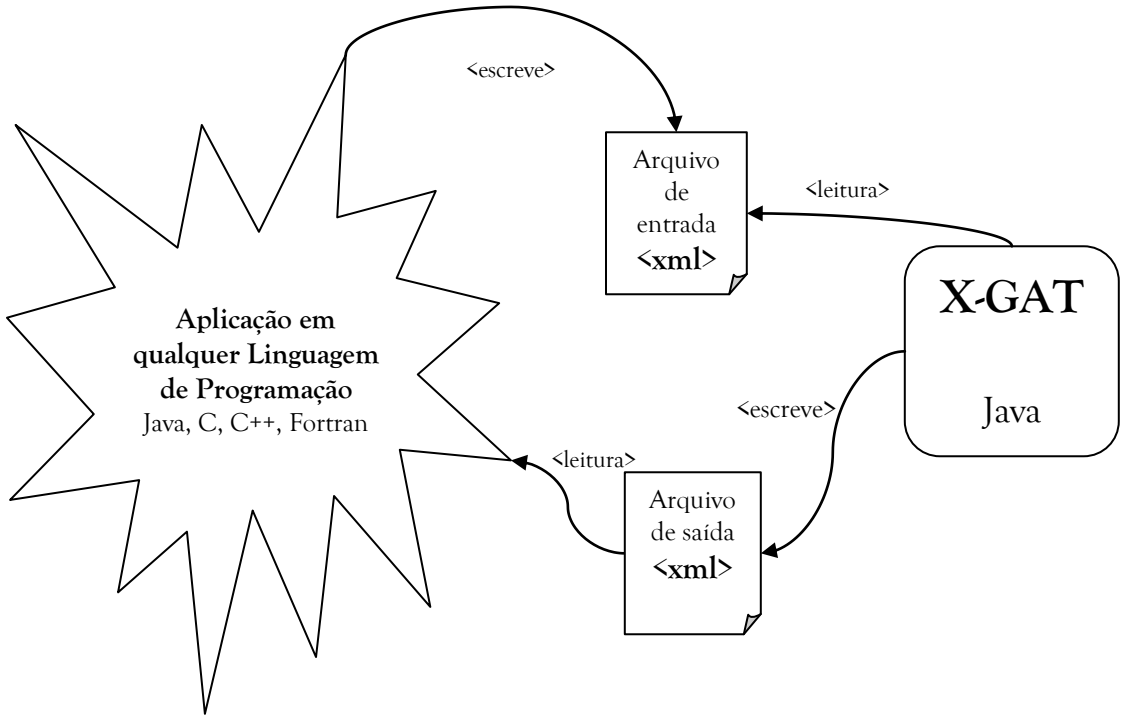


Figura 16 - Manipulação dos arquivos XML de entrada e saída para uma otimização.

Esta seção tem o objetivo de mostrar como uma simulação com AGs pode ser descrita através de *tags* em XML. Em XML, as *tags* são usadas para definir blocos de dados. Ela está organizada em duas subseções, uma que descreve o arquivo XML de entrada (Seção 3.5.1) e outra que descreve o arquivo XML de saída (Seção 3.5.2).

3.5.1 Descrição dos Dados de Entrada

O arquivo XML de entrada é responsável por descrever explicitamente as variáveis a serem otimizadas (genes), as funcionalidades a serem utilizadas na otimização e os dados que serão escritos no arquivo XML de saída. Esta seção detalha os componentes do arquivo XML de entrada, que podem ser vistos na Figura 17.

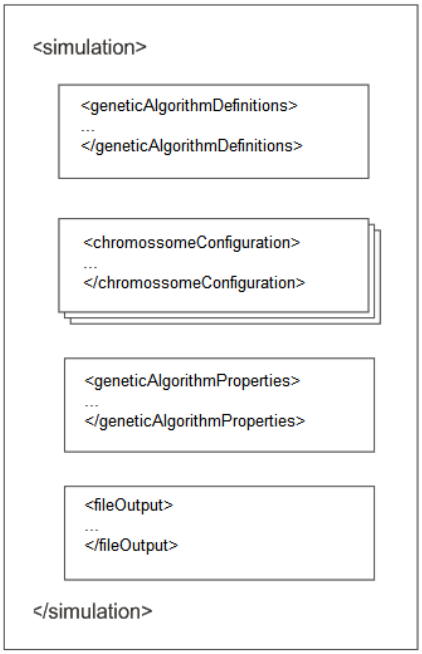


Figura 17 - Componentes do arquivo XML de entrada.

O arquivo XML de entrada é dividido em quatro componentes:

- **Genetic Algorithm Definitions:** este componente possui informações gerais sobre o AG a ser utilizado na otimização;
- **Chromossomes Configuration:** este componente possui os dados sobre os genes que compõem um cromossomo em uma otimização com AGs.
- **Genetic Algorithm Properties:** este componente possui informações sobre os dados comuns a qualquer AG e as funcionalidades a serem utilizadas no processo de otimização;
- **File Output:** Este componente possui informações sobre os dados a serem escritos no arquivo XML de saída;

Para cada um desses componentes, serão mostradas exatamente quais nomenclaturas devem ser adotadas nas *tags* para que a ferramenta X-GAT reconheça as informações provindas do arquivo XML de entrada.

3.5.1.1 O Componente *Genetic Algorithm Definitions*

O componente *Genetic Algorithm Definitions* deve conter quatro informações, organizadas na seqüência exibida pela Figura 18.

```
1    <geneticAlgorithmDefinitions>

2        <algorithm>ClassicGeneticAlgorithm</algorithm>
3    Ou
4        <algorithm>MiLambdaGeneticAlgorithm</algorithm>



5       <simulationName>SimulationTest</simulationName>

6        <dataType class="Real">
7            <precision>2</precision>
8        </dataType>
9    ou
10       <dataType class="Integer"/>
11   Ou
12       <dataType class="Char"/>

13       <optimization>Minimum</optimization>
14   Ou
15       <optimization>Maximum</optimization>

16   </geneticAlgorithmDefinitions>
```

Figura 18 - Descrevendo o componente *Genetic Algorithm Definitions*.

As *tags* das linhas 1 e 16 indicam onde começa e onde termina a descrição dos dados do componente *Genetic Algorithm Definitions*, respectivamente. A primeira informação a ser colocada no componente é qual o tipo de AG a ser executado. Na ferramenta X-GAT, existem dois AGs, já descritos na seção 3.3. O usuário deve escolher uma das linhas (2 ou 4) para explicitar qual o AG a ser utilizado na otimização. Na seqüência, o usuário deve dar um nome à simulação a ser realizada (linha 5). Após dar nome à simulação, o usuário deve dizer qual o tipo de dado de entrada que irá compor os genes de um indivíduo. Como mostrado na Figura 4, existem três tipos de dados que podem ser utilizados na ferramenta X-GAT (Integer, Real ou Char), que podem ser descritos pelas *tags* das linhas 6 a 12. Como pode ser visto na linha 6, o tipo de dado Real possui uma informação adicional, que é a precisão utilizada durante a execução da heurística de busca. Por último, o usuário deve informar se a otimização realizada visa à

minimização (linha 13) ou maximização (linha 15) do problema. Essa informação é necessária para ranquear os indivíduos de uma população, pois no caso de uma minimização, quanto menor o valor da nota de aptidão do indivíduo, maior é a chance deste indivíduo sobreviver e gerar descendentes. No caso da maximização, quanto maior a nota de aptidão de um indivíduo, maior é a chance deste indivíduo sobreviver e gerar descendentes.

3.5.1.2 O Componente *Chromossomes Configuration*

O componente *Chromossomes Configuration* deve conter a seqüência de informações sobre os genes que devem compor os indivíduos, organizadas da forma vista na Figura 19.

```
1   <chromossomesConfiguration class="realChromossomesConfiguration">
2    ou
3   <chromossomesConfiguration class="integerChromossomesConfiguration">
4    ou
5   <chromossomesConfiguration class="charChromossomesConfiguration">

6       <evaluationForm>JEPPopulationEvaluator</evaluationForm>
7       ou
8       <evaluationForm>org.ufpb.MinhaClasseDeAvaliacao</evaluationForm>

9       <description/>

10   <genesConfiguration>
11       <realTypeGene> ou <integerTypeGene> ou <charTypeGene>
12           <name>x</name>
13           <lowerBound>-2.0</lowerBound>
14           <upperBound>2.0</upperBound>
15        <realTypeGene/> ou <integerTypeGene/> ou <charTypeGene/>
16                                   .
                                     .
                                     .
17       <genesConfiguration/>

18   </chromossomesConfiguration>
```

Figura 19 – Descrevendo o componente *ChromossomesConfiguration*

As *tags* das linhas de 1 a 5 e a linha 18 indicam onde começa e onde termina a descrição do componente *ChromossomesConfiguration*. Primeiramente, o usuário deverá descrever qual classe de cromossomos vai ser utilizada na configuração. Note-se que esta parte deve estar de acordo com o tipo de classe escolhido na *tag* dataType, visto no componente *Genetic Algorithm Definitions*. Isto significa que, se o usuário escolheu o tipo de dado Real, a classe a ser utilizada é a classe *realChromossomesConfiguration*, vista na linha 1. As linhas 3 e 5 devem ser utilizadas para as classes Integer e Char, respectivamente. Em seguida, o usuário deve informar qual a classe de avaliação a ser utilizada na otimização. Caso o usuário deseje utilizar o JEP (linha 6, a *tag description* deve ser utilizada para descrever a função matemática a ser otimizada, podendo ser utilizadas as operações encontradas no Anexo A. Caso se deseje utilizar uma classe criada a partir da extensão da superclasse *PopulationEvaluationMethod*, o caminho completo da classe deve ser colocado na *tag* "<evaluationForm>" da forma vista na linha 8. O caminho completo dos pacotes é necessário para que se faça a instanciação dinâmica da nova classe criada, através da API de reflexão da linguagem Java. Por último, deverão ser descritas as informações sobre os genes a serem utilizados na otimização (linhas de 10 a 17). A descrição de um gene começa com a *tag* "<genesConfiguration>", e posteriormente deve ser utilizada a *tag* correspondente a classe do tipo de dado escolhido no início do componente. No exemplo da Figura 19, é mostrada a descrição de um gene do tipo Real (linhas de 11 a 15), utilizado no caso de ter sido utilizada a classe *realChromossomesConfiguration* na linha 1 do arquivo XML. Serão descritas três informações: nome da variável (linha 12); limite inferior (linha 13); limite superior (linha 14). Os mesmos dados são descritos para os tipos Integer e Char. Quanto ao tipo Char, os limites inferiores e superiores devem ser baseados na tabela ASCII. As linhas de 11 a 15 devem ser repetidas caso mais de um gene seja utilizado na otimização, descrevendo-se assim seqüencialmente as variáveis de entrada do problema a ser resolvido.

3.5.1.3 O Componente Genetic Algorithm Properties

O componente *Genetic Algorithm Properties* deve conter as informações sobre as funcionalidades utilizadas pelo AG. As funcionalidades que podem ser configuradas, já explicadas na seção 3.4, podem ser descritas pelas *tags* mostradas na Figura 20.

```
1   <geneticAlgorithmProperties>
2       <quantityOfIndividuals>200</quantityOfIndividuals>
3       <elitism>true</elitism> ou <elitism>false</elitism>
4       <initializationMethod>GridInitialization</initializationMethod>
5       ou
6       <initializationMethod>RandomInitialization</initializationMethod>
7       <selectionAlgorithm>RouletteWheel</selectionAlgorithm>
8       ou
9       <selectionAlgorithm>TournamentSelection</selectionAlgorithm>
10      <crossoverMethod>AverageCrossover</crossoverMethod>
11      ou
12      <crossoverMethod>BLXAlphaCrossover</crossoverMethod>
13      ou
14      <crossoverMethod>LinearCrossover</crossoverMethod>
15      <mutationMethod>UniformMutation</mutationMethod>
16      ou
17      <mutationMethod>LimitMutation</mutationMethod>
18      <rate>
19          <crossover>80.0</crossover>
20          <mutation>2.0</mutation>
21      </rate>
23          <stopCondition class="Convergence">
24          <convergencePercentage>80.0</convergencePercentage>
25          <quantityOfequalBestSets>15</quantityOfequalBestSets>
26      </stopCondition>
27      ou
28      <stopCondition class="FiniteNumberOfGenerations">
29          <numberOfGenerations>10</numberOfGenerations>
30      </stopCondition>
31      ou
32      <stopCondition class="FunctionOptimalValue">
33          <value>3.</value>
34      </stopCondition>
35      ou
36      <stopCondition class="LessOrEqualTo">
37                                  ou <stopCondition class="GreaterOrEqualTo">
38          <value>3.2</value>
39      </stopCondition>
40  </geneticAlgorithmProperties>
```

Figura 20 - Descrevendo o componente *Genetic Algorithm Properties*.

As *tags* das linhas 1 e 40 indicam o começo e o fim do componente *Genetic Algorithm Properties*. A primeira informação a ser descrita é a quantidade de indivíduos por geração (linha 2), onde deve ser colocado um número inteiro com a quantidade desejada. Em seguida, deve ser informado se a técnica de elitismo deve ser usada (detalhada na Seção 2.2.3) por meio de um valor booleano (linha 3). Depois, devem ser informados: método de inicialização (Seção 3.4.1) a ser adotado para gerar a primeira população de indivíduos (linhas de 4 a 6); o método de seleção (Seção 3.4.2) a ser usado para escolher os melhores indivíduos para que as técnicas de reprodução sejam aplicadas (linhas de 7 a 9); o método de *crossover* (Seção 3.4.3) a ser usado para trocar informações entre dois indivíduos com o objetivo de obter um novo indivíduo para a próxima geração (linhas de 10 a 14); e o método de mutação (Seção 3.4.4) a ser aplicado em cada novo indivíduo gerado (linhas de 15 a 17). Na seqüência, as taxas de *crossover* e mutação devem ser informadas (linhas de 18 a 21). A última funcionalidade a ser descrita é qual o critério de parada (Seção 3.4.6) utilizado para finalizar o processo de otimização com AGs (linhas de 23 a 39). Caso o usuário deseje parar o AG utilizando o critério de convergência (linhas de 23 a 26), ele deve informar dois valores: o primeiro (linha 24) indica a porcentagem de indivíduos que contém genes iguais (fator de convergência primária descrito na Seção 2.2.3); e o segundo (linha 25) indica que, se forem evoluídas essa quantidade de gerações e não houver melhoras no valor da nota de aptidão, o AG deve ser parado. Alternativamente, se for adotado o critério de quantidade finita de gerações, deve ser apenas informado o valor dessa quantidade de gerações a serem evoluídas (linhas de 28 a 30). O critério de parada descrito pelas linhas de 32 a 34 (valor ótimo do problema de otimização) apenas recebe como informação o valor exato do ótimo global a ser encontrado. O último critério de parada (linhas de 36 a 39) é o que compara um valor menor (linha 36) ou maior (linha 37) com os valores computados pela função de aptidão de um indivíduo. Se o valor encontrado por um indivíduo em uma geração (*t*) for menor (ou maior) que o número descrito na *tag* “<value>” (linha 38) o AG deve parar nesta geração (*t*).

3.5.1.4 O Componente File Output

O componente *File Output* contém dados necessários para a geração do arquivo XML de saída da ferramenta X-GAT. Os dois dados que devem ser informados nesse componente podem ser vistos na Figura 21.

```
1    <fileOutput>

2        <name>outputFileForXGAT.xml</name>
3        <interval>5</interval>

4    </fileOutput>
```

Figura 21 - O Componente *File Output*.

As linhas 1 e 4 indicam o início e o fim do componente *File Output*. Primeiramente, o usuário deverá informar o nome do arquivo XML de saída a ser gerado ao final do processo de otimização com um AG (linha 2). Esse arquivo deve necessariamente conter a extensão *.xml*. Em seguida, o usuário deve informar de quantas em quantas gerações, os dados das populações evoluídas devem ser armazenados no arquivo XML de saída (linha 3). No exemplo da Figura 21, onde se encontra o valor 5, informações sobre os indivíduos das gerações (1, 5, 10... até a última geração evoluída) estarão disponíveis no arquivo XML de saída, podendo estes dados ser utilizados para verificar como foi dada a convergência para o melhor valor encontrado, ou então para saber exatamente quais áreas do espaço de busca foram realmente exploradas.

3.5.2 Descrição dos Dados de Saída

O arquivo XML de saída deve conter as informações sobre a melhor solução encontrada pelo AG executado e também fornecer dados que auxiliem o usuário a avaliar as regiões exploradas do espaço de busca durante a execução da otimização. Esta seção detalha os componentes do arquivo XML de saída da ferramenta X-GAT, que podem ser vistos na Figura 22.

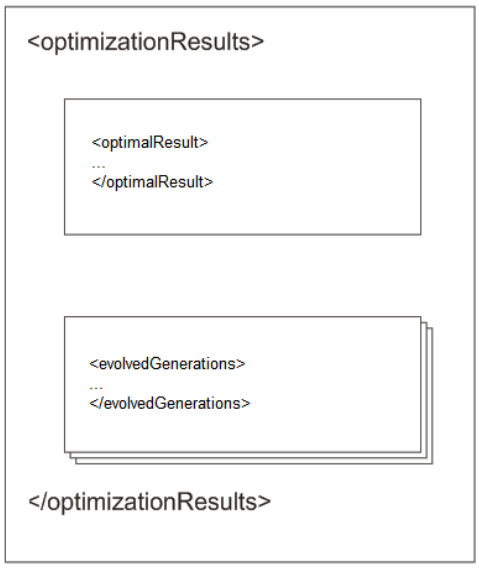


Figura 22 - Componentes do arquivo XML de saída.

O arquivo XML de saída possui dois componentes:

- **Optimal Result:** este componente contém dados com informações a respeito do melhor valor encontrado pela execução do AG;
- **Evolved Generations:** este componente contém informações sobre os indivíduos evoluídos durante todo o processo de otimização realizado.

Para cada um desses componentes, serão mostrados os significados das *tags* para uma melhor compreensão do arquivo XML de saída.

3.5.2.1 O Componente *Optimal Result*

O componente *Optimal Result* contém informações sobre o valor ótimo encontrado pela execução de um AG. A Figura 23 mostra cada *tag* gerada no arquivo de saída. São fornecidas neste arquivo XML de saída quatro informações sobre a execução da busca: o melhor valor encontrado para cada gene (parâmetro de entrada do problema de otimização), que pode ser visto na linha 2, ordenados pela mesma seqüência descrita no componente *ChromossomeConfiguration* (Seção 3.5.1.2); o valor da nota de aptidão do melhor indivíduo, que pode ser visto na linha 3; a quantidade gerações evoluídas, que pode ser vista na linha 4; e finalmente, o tempo de total de execução em segundos, que pode ser visto na linha 5.

```
1    <optimalResult>
2        <bestSolution>0.02,-0.98</bestSolution>
3        <optimalValue>3.18</optimalValue>
4        <quantityOfGenerations>20</quantityOfGenerations>
5        <elapsedTime>40</elapsedTime>
6    </optimalResult>
```

Figura 23 - O Componente *Optimal Result*.

3.5.2.2 O Componente *Evolved Generations*

O componente *Evolved Generations* contém dados sobre as gerações evoluídas durante a execução da otimização com AG. Esses dados são interessantes por duas razões: (1) pode ser feita uma análise de como o AG se comportou até chegar à solução encontrada; e (2) os dados mostram quais regiões do espaço de busca foram realmente analisadas pela execução do AG.

Este componente contém duas informações, como pode ser visto na Figura 24. As linhas 1 e 26 indicam o início e o final do componente *Evolved Generations*. A linha 2 informa quantos indivíduos foram utilizados em cada geração evoluída. As linhas de 3 a 25 mostram informações sobre cada conjunto de indivíduos que formaram uma população. A linha 4 marca o início da descrição de uma população, seguido na linha 5 do identificador dessa população. Neste exemplo, a linha 5 denota a primeira população usada pelo AG e a linha 20 mostra a segunda população evoluída pela heurística. Em cada uma dessas populações, são mostrados todos os cromossomos (linhas de 6 a 17), explicitando exatamente quais os valores dos seus genes (linhas 9 e 10) e o valor da nota de aptidão (linha 12).

```
1   <evolvedGenerations>
2       <quantityOfIndividualsPerPopulation>200</quantityOfIndividualsPerPopulation>
3           <populationSet>
4               <population>
5           <id>1</id>
6                       <chromossomesSet>
7                           <chromossome>
8                               <genes>
9                                   <double>0.1</double>
10                                  <double>-1.1</double>
11                              </genes>
12                              <fitness>14.68</fitness>
13                          </chromossome>
14                          <chromossome>
15                              (...)
16                          </chromossome>
17                      </chromossomesSet>
18              </population>
19              <population>
20                  <id>2</id>
21                  (...)
22              </population>
23              (...)
24              </population>
25          </populationSet>
26  <evolvedGenerations>
```

Figura 24 - O componente *Evolved Generations*.

3.6 Considerações finais

Neste capítulo, foi apresentado o processo de modelagem e detalhes da construção da ferramenta X-GAT. Após a identificação das entidades necessárias à implementação do projeto, que foram baseadas nas questões levantadas no capítulo anterior (Capítulo 2), foram mostradas as técnicas de engenharia de software utilizadas neste processo de modelagem e construção. Em seguida, foram mostrados em detalhes

os AGs e as funcionalidades implementadas pelo X-GAT. Por último, foram mostrados os formatos de descrever os arquivos XML (disposição das *tags* em XML para uso de cada funcionalidade) de entrada e saída.

Após o processo de modelagem e implementação da ferramenta de otimização X-GAT, tornou-se necessário validá-la em diversos problemas, com o propósito de mostrar que os objetivos enumerados na Seção 1.2 foram atendidos. O capítulo seguinte (Capítulo 4) mostra algumas aplicações nas quais a ferramenta X-GAT foi aplicada, sendo também avaliados os resultados de cada uma dessas aplicações segundo métricas específicas de cada problema.

Capítulo

4

Testes e Resultados

Este capítulo apresenta testes para a validação da ferramenta de otimização X-GAT. Primeiramente, a ferramenta de otimização é testada em algumas funções matemáticas nas quais o valor do ótimo global é conhecido, tendo como principal objetivo verificar se a implementação é capaz de encontrar este valor ótimo (Seção 4.1). Em seguida, a ferramenta é aplicada em otimização dos parâmetros de modelos hidrológicos, que são muito comuns na área de hidrologia (Seção 4.2). Por último, a ferramenta é aplicada na área de trajetórias de objetos móveis (Seção 4.3) para otimizar a escolha dos valores dos parâmetros de entrada dos algoritmos de clusterização espacial.

4.1 Otimização de Funções Matemáticas

Nesta seção, será apresentada a otimização de quatro funções matemáticas que possuem solução conhecida. Implementar uma ferramenta de otimização e aplicá-la em problemas com soluções conhecidas com o objetivo de validá-la é uma prática comum, como pode ser visto nos trabalhos de Santos *et*. al. (2003), Madeiro, (2010), Christiani (2009) e Hrstka & Kucerová (2004). O principal objetivo deste experimento é verificar primeiramente se a ferramenta X-GAT é capaz de encontrar ótimos globais em problemas de otimização, já que para cada uma das quatro funções este valor ótimo é conhecido. Se a ferramenta X-GAT for capaz de encontrar estes ótimos, então concluímos que existe possibilidade de que em outros problemas, a ferramenta X-GAT poderá ser capaz de encontrar o valor ótimo do problema. Foram criadas diversas configurações para cada uma dessas quatro funções matemáticas, e os resultados foram comparados com o objetivo de analisar o comportamento do AG em cada uma das configurações, além de verificar se o ótimo global foi encontrado exatamente.

As quatro funções utilizadas nesta primeira fase de testes são detalhadas na Quadro 3. Os valores das variáveis otimizadas e do ótimo global a ser encontrado em cada um destes problemas de otimização podem ser vistos na última coluna da Quadro 3.

Quadro 3 - Descrição das funções matemáticas utilizadas nos testes.

Nome da função	Espaço de Busca	Função	Ótimo Global
Goldstein-Price	x[-2, 2] y[-2, 2]	**f** = (1. + (((x + y + 1.0)^2) * (19. - 14.*x + 3.*x^2 - 14.*y + 6.*x*y +3.*y^2))) * (30. + (((2.*x - 3.*y)^2) * (18. - 32.*x + 12.*x^2 + 48.*y -36.*x*y + 27.*y^2)))	**(x,y)** = (0, -1); **f(x,y)** = 3.
Rosenbrock	x[-2.048, 2.048] y[-2.048, 2.048]	**f** = (100*(y - x^2)^2+(1-x)^2)	**(x,y)** = (1, 1); **f(x,y)** = 0.
Six-hump Camelback	x[-3, 3] y[-2, 2]	**f** = ((4. - 2.1*x^2 + x^4/3.)*x^2+x*y + (-4. + 4.*y^2)*y^2)	**(x,y)** = (-0.0898, 0.7126) ou (0.0898, -0.7126) ; **f(x,y)** = -1.0316.
Rastringin	x[-1, 1] y[-1, 1]	**f** = (x^2 + y^2 - cos(18.0*x) - cos(18.0*y))	**(x,y)** = (0, 0) **f(x,y)** = 0.0

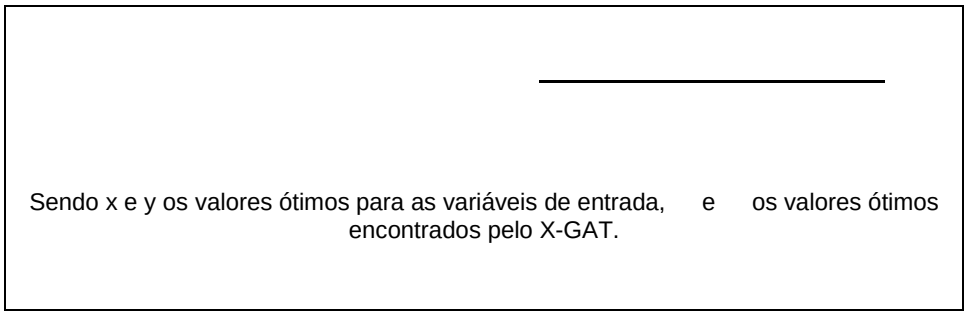
Cada uma destas funções matemáticas foi submetida a um processo de otimização utilizando as configurações mostradas na Quadro 4. Para cada um dos dois AGs implementados, mostrados na Seção 3.3, foram executadas essas 24 configurações, com um total de 48 simulações realizadas para cada função matemática. Os parâmetros de entrada: taxa de *crossover*, taxa de mutação, critério de parada e forma de avaliação foram fixados. Para as taxas de *crossover* e mutação foram utilizadas taxas baseadas no trabalho de (De Jong, 1975), com os valores de 80% para *crossover* e 3.5% para mutação. Como critério de parada foi escolhido uma quantidade finita de gerações (100 gerações evoluídas) para terminar o AG. Este critério foi fixado para que todas as execuções tivessem a mesma quantidade de iterações (evoluções), proporcionando assim, condições iguais a cada uma das configurações para que o valor do ótimo global fosse encontrado. Como forma de avaliação foi utilizada a classe *JEPEvaluationMethod*, previamente descrita na Seção 3.4.5.

Quadro 4 - Configurações utilizadas por cada AG.

Configuração	Seleção	Inicialização	*Crossover*	Mutação	Tamanho da População
conf_1	Torneio	Randômica	Média	Limite	50
conf_2	Torneio	Randômica	Média	Limite	100
conf_3	Torneio	Randômica	BLX-alpha	Uniforme	50
conf_4	Torneio	Randômica	BLX-alpha	Uniforme	100
conf_5	Torneio	Randômica	Linear	Uniforme	50
conf_6	Torneio	Randômica	Linear	Uniforme	100
conf_7	Torneio	Grid	Média	Limite	50
conf_8	Torneio	Grid	Média	Limite	100
conf_9	Torneio	Grid	BLX-alpha	Uniforme	50
conf_10	Torneio	Grid	BLX-alpha	Uniforme	100
conf_11	Torneio	Grid	Linear	Uniforme	50
conf_12	Torneio	Grid	Linear	Uniforme	100
conf_13	Roleta	Randômica	Média	Limite	50
conf_14	Roleta	Randômica	Média	Limite	100
conf_15	Roleta	Randômica	BLX-alpha	Uniforme	50
conf_16	Roleta	Randômica	BLX-alpha	Uniforme	100
conf_17	Roleta	Randômica	Linear	Uniforme	50
conf_18	Roleta	Randômica	Linear	Uniforme	100
conf_19	Roleta	Grid	Média	Limite	50
conf_20	Roleta	Grid	Média	Limite	100
conf_21	Roleta	Grid	BLX-alpha	Uniforme	50
conf_22	Roleta	Grid	BLX-alpha	Uniforme	100
conf_23	Roleta	Grid	Linear	Uniforme	50
conf_24	Roleta	Grid	Linear	Uniforme	100

Para comparar os resultados obtidos nestas 48 simulações com AGs, foram adotadas as seguintes métricas: (1) precisão, quando o ótimo global for achado com exatidão (representado pelo símbolo ‘o’ quando o valor for encontrado e pelo símbolo ‘x’ quando o valor não for encontrado); (2) Distância euclidiana, quanto o valor do ótimo local encontrado se aproxima do valor ótimo global real da função, utilizando a Equação 5; (3) tempo de processamento em segundos.

Equação 5 – Distância euclidiana entre o valor ótimo da função e o valor encontrado como ótimo pelo X-GAT.



Sendo x e y os valores ótimos para as variáveis de entrada, e os valores ótimos encontrados pelo X-GAT.

Mostramos na Figura 25, como a ferramenta X-GAT é configurada por meio de um arquivo XML. Utilizamos como exemplo a configuração 1 definida pela Quadro 4 para a função de Rastrigin. Para cada uma das configurações adotadas, as descrições das funcionalidades utilizadas devem utilizar as *tags* definidas na Seção 3.5.1.

```
<!-- X-2GA Version 1.0 -->
- <simulation>
  - <geneticAlgorithmDefinitions>
      <algorithm>ClassicGeneticAlgorithm</algorithm>
      <simulationName>SimulationTest1</simulationName>
    - <dataType class="Real">
        <precision>4</precision>
      </dataType>
      <optimization>Minimum</optimization>
    </geneticAlgorithmDefinitions>
  - <chromossomesConfiguration class="realChromossomesConfiguration">
      <evaluationForm>JEPPopulationEvaluator</evaluationForm>
      <description>(x^2 + y^2 - cos(18.0*x) - cos(18.0*y))</description>
      <constraints/>
    - <genesConfiguration>
      - <realTypeGene>
          <name>x</name>
          <lowerBound>-1.0</lowerBound>
          <upperBound>1.0</upperBound>
        </realTypeGene>
      - <realTypeGene>
          <name>y</name>
          <lowerBound>-1.0</lowerBound>
          <upperBound>1.0</upperBound>
        </realTypeGene>
      </genesConfiguration>
    </chromossomesConfiguration>
  - <geneticAlgorithmProperties>
      <quantityOfIndividuals>50</quantityOfIndividuals>
      <elitism>true</elitism>
      <initializationMethod>RandomInitialization</initializationMethod>
      <selectionAlgorithm>TournamentSelection</selectionAlgorithm>
      <crossoverMethod>AverageCrossover</crossoverMethod>
      <mutationMethod>LimitMutation</mutationMethod>
    - <rate>
        <crossover>80.0</crossover>
        <mutation>3.5</mutation>
      </rate>
    - <stopCondition class="FiniteNumberOfGenerations">
        <numberOfGenerations>100</numberOfGenerations>
      </stopCondition>
```

Figura 25 - Exemplo de um arquivo XML de entrada, utilizado para a configuração 1 da função de Rastrigin.

4.1.1 A Função Goldstein-Price

O problema de Goldstein-Price (Goldstein & Price 1971) é uma função para teste de otimização global. Sua dificuldade é que seu valor mínimo rapidamente decresce, mas chega a uma superfície quase plana, onde se localiza este valor mínimo, tornando assim difícil encontrá-lo com precisão. A função de Goldstein-Price pode ser vista na Figura 26.

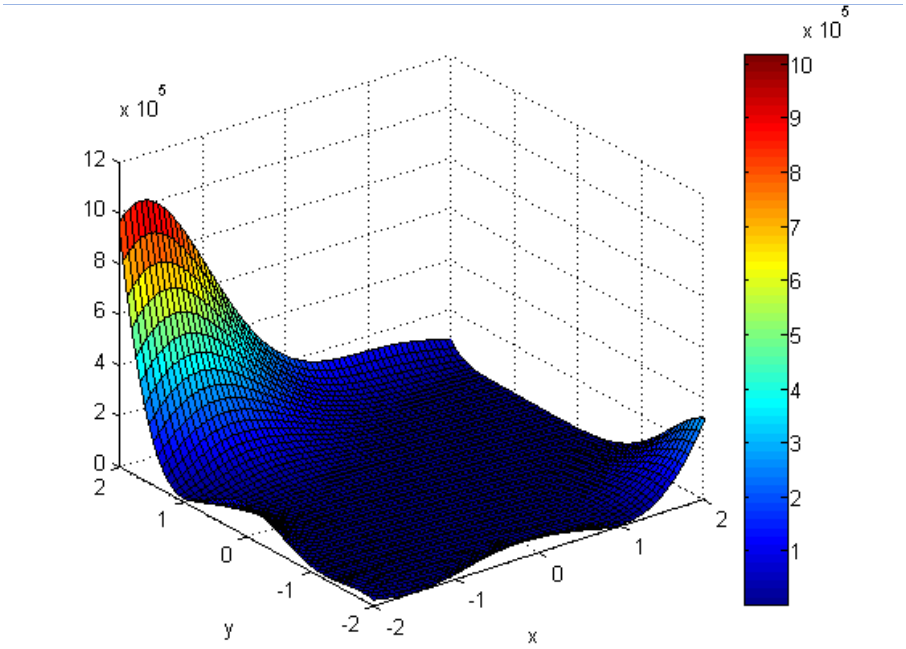


Figura 26 - A função Goldstein-Price.

A Tabela 1 mostra os resultados obtidos pela execução das configurações descritas no Quadro 4 para o AG clássico. Como pode ser visto na Tabela 1, algumas soluções não foram capazes de encontrar o ótimo global da função de Goldstein-price. As configurações de 1 a 6 utilizam como forma de inicialização, a inicialização randômica e como método de seleção, a seleção por torneio. Essas duas sendo utilizadas em conjunto não conseguiram encontrar o ótimo global da função. Utilizando a seleção por torneio apenas as configurações 11 e 12 conseguiram encontrar o ótimo, tendo estas duas sido inicializadas por *grid*. As configurações que utilizaram a seleção por roleta obtiveram melhores resultados, se comparada com a seleção por torneio (configurações 15 a 18, e de 21 a 24). Comparando-se os métodos de *crossover*, o que obteve melhor desempenho foi o linear, como pode ser visto nas configurações 11, 12, 17, 18, 23, 24. O *crossover* BLX-alpha também obteve bons resultados (configurações 15, 16, 21 e 22). Apenas uma configuração que utiliza o *crossover* média (configuração 7), conseguiu achar o ótimo, sendo esta inicializada por *grid*.

Tabela 1 - Resultados de otimização da função de Goldstein-Price para o AG clássico.

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima	0.0000, -1.0000	3.0000	O	0.0000	xx
conf_1	-0.0278, -0.9103	6.9628	X	0.0939	3s
conf_2	0.0121, -0.9082	6.787	X	0.0925	5s
conf_3	0.1309, -0.9593	7.3722	X	0.137	5s
conf_4	0.0421, -0.9586	3.7883	X	0.0590	5s
conf_5	-0.0135, -0.947	4.3467	X	0.0546	9s
conf_6	0.0243, -0.9776	3.2427	X	0.0330	18s
conf_7	0.0000, -1.0000	3.0000	O	0.0000	4s
conf_8	0.1111, -0.889	8.7566	X	0.1570	6s
conf_9	-0.0769, -1.0006	4.6122	X	0.0769	4s
conf_10	-0.0738, -0.9273	6.8903	X	0.1035	6s
conf_11	0.0000, -1.0000	3.0000	O	0.0000	10s
conf_12	0.0000, -1.0000	3.0000	O	0.0000	19s
conf_13	0.0000, -0.9981	3.0013	X	0.0021	3s
conf_14	0.0000, -0.9993	3.0002	X	0.0011	5s
conf_15	0.0000, -1.0000	3.0000	O	0.0000	3s
conf_16	0.0000, -1.0000	3.0000	O	0.0000	5s
conf_17	0.0000, -1.0000	3.0000	O	0.0000	9s
conf_18	0.0000, -1.0000	3.0000	O	0.0000	19s
conf_19	-0.0064, -0.9947	3.0293	X	0.0083	3s
conf_20	-0.0017, -1.0024	3.0023	X	0.0029	5s
conf_21	0.0000, -1.0000	3.0000	O	0.0000	19s
conf_22	0.0000, -1.0000	3.0000	O	0.0000	5s
conf_23	0.0000, -1.0000	3.0000	O	0.0000	9s
conf_24	0.0000, -1.0000	3.0000	O	0.0000	18s

A Tabela 2 mostra os resultados obtidos pela execução das configurações descritas na Quadro 4 para o AG (μ + λ). Como pode ser visto na Tabela 2, as configurações que adotaram a inicialização randômica não obtiveram êxito para encontrar o ótimo global. Isso também aconteceu com as configurações que adotaram a seleção por torneio como método de seleção de indivíduos. Quando comparados os

resultados da Tabela 1 com os da Tabela 2 podemos ver que o AG clássico achou mais vezes o ótimo global (11 vezes) que o AG (μ+λ) (8 vezes).

Tabela 2 - Resultados de otimização da função de Goldstein-Price para o AG (μ+λ).

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima	0.0000, -1.0000	3.0000	O	0.0000	xx
conf_1	0.0559, -0.9022	7.2438	X	0.1126	3s
conf_2	-0.1129, -0.9793	7.3066	X	0.1147	6s
conf_3	0.0240, -0.8898	8.7138	X	0.112	6s
conf_4	-0.0245, -0.9577	4.0531	X	0.0488	6s
conf_5	0.0330, -0.9396	4.4750	X	0.0688	10s
conf_6	0.0168, -0.9623	3.5513	X	0.0412	20s
conf_7	-0.0018, -0.9131	6.4426	X	0.0869	3s
conf_8	0.1109, -0.8889	8.7651	X	0.1569	5s
conf_9	-0.0525, -1.0475	4.2122	X	0.0707	3s
conf_10	-0.1124, -0.9836	7.1646	X	0.1135	6s
conf_11	0.0000, -1.0000	3.0000	O	0.0000	9s
conf_12	0.0000, -1.0079	3.0273	X	0.0079	18s
conf_13	-0.0017, -1.001	3.0007	X	0.0019	3s
conf_14	-0.0143, -1.0094	3.0614	X	0.0171	5s
conf_15	0.0000, -1.0000	3.0000	O	0.0000	3s
conf_16	0.0000, -0.9999	3.0001	X	0.0001	5s
conf_17	0.0000, -1.0000	3.0000	O	0.0000	10s
conf_18	0.0000, -1.0000	3.0000	O	0.0000	18s
conf_19	0.0000, -1.002	3.0021	X	0.0002	3s
conf_20	0.0000, -0.9994	3.0001	X	0.0006	5s
conf_21	0.0000, -1.0000	3.0000	O	0.0000	3s
conf_22	0.0000, -1.0000	3.0000	O	0.0000	5s
conf_23	0.0000, -1.0000	3.0000	O	0.0000	9s
conf_24	0.0000, -1.0000	3.0000	O	0.0000	19s

4.1.2 A Função Rosenbrock

A função de Rosenbrock (Rosenbrock, 1960) é um problema clássico de otimização. O ótimo global é encontrado em um vale longo, estreito, chato e parabólico, como pode ser visto na Figura 27. Achar este vale não é uma tarefa difícil, entretanto a convergência para o mínimo global é difícil.

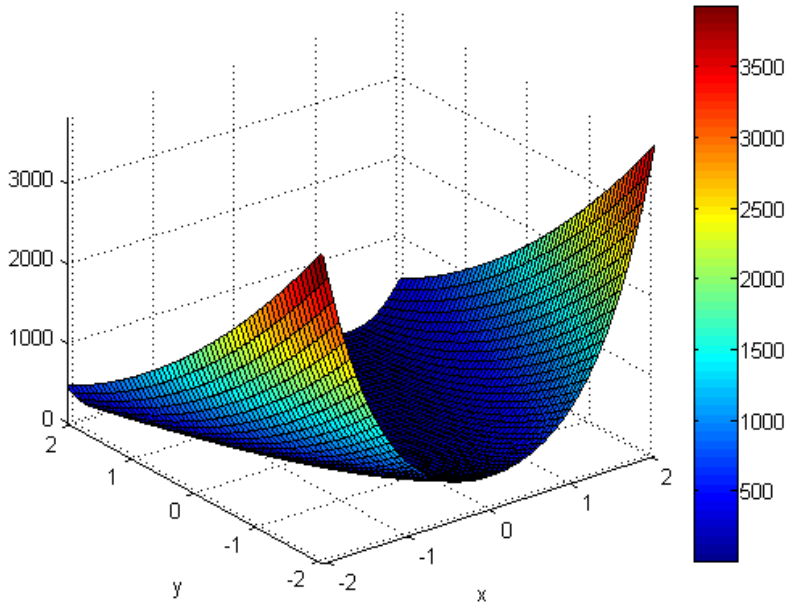


Figura 27 - A função Rosenbrock.

A Tabela 3 mostra os resultados obtidos pela execução das configurações descritas no Quadro 4 para o AG clássico a fim de otimizar a função de Rosenbrock. Como pode ser visto na Tabela 3, as configurações que encontraram o valor ótimo (configuracoes 17, 18, 23 e 24) da função Rosenbrock utilizaram o *crossover* linear e o método de seleção por roleta, independentemente da forma de inicialização.

Tabela 3 - Resultados de otimização da função de Rosenbrock para o AG clássico.

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima	1.0000, 1.0000	0.0000	O	0.0000	xx
conf_1	0.7847, 0.5993	0.0734	X	0.4548	2s
conf_2	0.8162, 0.6511	0.0565	X	0.3943	3s
conf_3	0.8318, 0.6358	0.3429	X	0.4011	4s
conf_4	0.8975, 0.8464	0.1777	X	0.1846	4s
conf_5	1.2748, 1.6096	0.0995	X	0.6686	4s
conf_6	0.8351, 0.701	0.0284	X	0.3414	7s
conf_7	0.0000, 0.0000	1.0000	X	1.4143	2s
conf_8	0.9101, 0.9102	0.6791	X	0.1270	3s
conf_9	0.0975, 0.0243	0.8363	X	1.3290	2s
conf_10	0.7971, 0.5700	0.4684	X	0.4754	3s
conf_11	1.0237, 1.0237	0.0594	X	0.0335	4s
conf_12	0.6825, 0.455	0.1124	X	0.6307	7s
conf_13	0.7008, 0.501	0.0992	X	0.5818	2s
conf_14	0.939, 0.8837	0.0041	X	0.1313	3s
conf_15	0.8596, 0.741	0.0201	X	0.2946	2s
conf_16	0.7471, 0.5669	0.0715	X	0.5015	3s
conf_17	1.0000, 1.0000	0.0000	O	0.0000	4s
conf_18	1.0000, 1.0000	0.0000	O	0.0000	7s
conf_19	0.7196, 0.5178	0.0786	X	0.5578	2s
conf_20	0.852, 0.7309	0.0244	X	0.3071	3s
conf_21	0.8794, 0.7679	0.0175	X	0.2615	2s
conf_22	0.8876, 0.7923	0.0146	X	0.2361	3s
conf_23	1.0000, 1.0000	0.0000	O	0.0000	4s
conf_24	1.0000, 1.0000	0.0000	O	0.0000	7s

A Tabela 4 mostra os resultados obtidos pela execução das configurações descritas na Quadro 4 para o AG (μ + λ). Como pode ser visto na Tabela 4, as configurações 17, 18, 23 e 24 também encontraram o mínimo global, resultado idêntico ao que aconteceu com o AG clássico mostrado anteriormente.

Tabela 4 - Resultados de otimização da função de Rosenbrock para o AG (μ+λ).

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima	1.0000, 1.0000	0.0000	O	0.0000	xx
conf_1	1.0222, 1.0402	0.0026	X	0.0459	2s
conf_2	0.3288, 0.0599	0.6829	X	1.1551	3s
conf_3	0.5917, 0.3449	0.1694	X	0.7719	4s
conf_4	0.9579, 0.937	0.0395	X	0.0757	4s
conf_5	0.3068, 0.0877	0.4846	X	1.1457	4s
conf_6	0.4742, 0.2191	0.2797	X	0.9414	7s
conf_7	0.0000, 0.0000	1.0	X	1.4142	2s
conf_8	0.6826, 0.455	0.1127	X	0.6306	4s
conf_9	0.6088, 0.3730	0.1535	X	0.7390	2s
conf_10	0.1069, 0.0090	0.7982	X	1.3340	4s
conf_11	1.0235, 1.024	0.0560	X	0.0335	4s
conf_12	1.024, 1.0512	0.0012	X	0.0565	7s
conf_13	0.7283, 0.5263	0.0755	X	0.5460	2s
conf_14	0.5953, 0.3672	0.1802	X	0.7511	3s
conf_15	0.7605, 0.5794	0.0574	X	0.4840	2s
conf_16	1.0197, 1.0329	0.0051	X	0.0383	3s
conf_17	1.0000, 1.0000	0.0000	O	0.0000	4s
conf_18	1.0000, 1.0000	0.0000	O	0.0000	7s
conf_19	0.3145, 0.0854	0.4881	X	1.1429	2s
conf_20	0.9153, 0.8385	0.0072	X	0.1823	3s
conf_21	1.037, 1.0766	0.0015	X	0.0850	2s
conf_22	0.7552, 0.5864	0.0857	X	0.4806	3s
conf_23	1.0000, 1.0000	0.0000	O	0.0000	4s
conf_24	1.0000, 1.0000	0.0000	O	0.0000	7s

4.1.3 A Função Six-hump Camelback

A função Six-hump Camelback (Dixon & Szego, 1978) é classificada como de difícil solução porque ela é contínua, não convexa, multimodal (possui 6 mínimos locais,

dos quais apenas dois são globais) e de baixa dimensão, como pode ser visto na Figura 28.

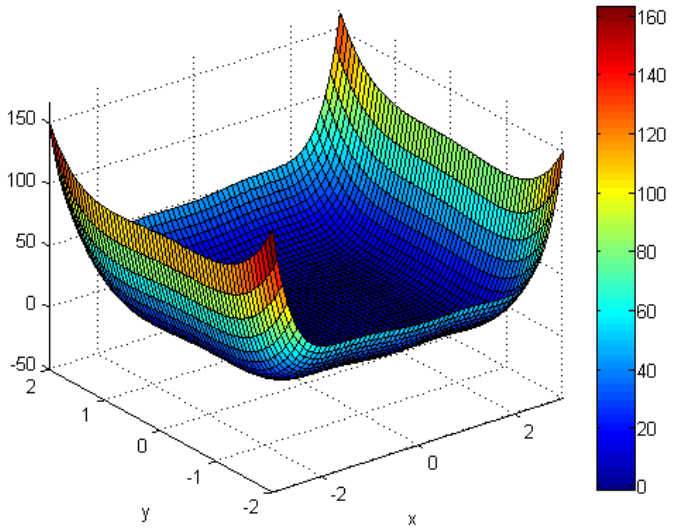


Figura 28 - A função Six-hump Camelback.

A Tabela 5 mostra os resultados obtidos pela execução das configurações descritas na Quadro 4 para o AG clássico. A função Sixhump-Camelback possui dois ótimos globais (Ótimo 1 e Ótimo 2 na Tabela 5). A distância exibida na Tabela 5 é a menor distancia para um destes valores ótimos. Para a função Sixhump-Camelback não houve predominância de funcionalidades que encontraram o ótimo global para a busca com o AG clássico. A configuração 3, como pode ser vista na Tabela 5, encontra o valor do ótimo global. Entretanto, os parâmetros de entrada não são os ótimos, havendo assim uma distância pontual de 0.0034 para o ótimo global exato a ser encontrado no problema. Dessa forma, assumimos este resultado como o de que o ótimo global não foi encontrado. Assim como aconteceu nos testes anteriores, as configurações 23 e 24 permanecem sempre encontrando exatamente o ótimo global.

Tabela 5 - Resultados de otimização da função Sixhump-Camelback para o AG clássico.

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima 1	-0.0898, 0.7126	-1.0316	O	0.0000	xx
Ótima 2	0.0898, -0.7126	-1.0316	O	0.0000	xx
conf_1	0.0281, -0.6964	-1.0156	X	0.0637	2s
conf_2	-0.0938, 0.7267	-1.0300	X	0.0146	4s
conf_3	0.0873, -0.7102	-1.0316	X	0.0034	4s
conf_4	-0.0786, 0.7083	-1.0311	X	0.0119	4s
conf_5	-0.0734, 0.7059	-1.0304	X	0.0177	5s
conf_6	-0.0673, 0.7002	-1.0287	X	0.0256	10s
conf_7	0.0625, -0.7501	-1.0156	X	0.0463	2s
conf_8	0.0833, -0.7085	-1.0314	X	0.0076	4s
conf_9	-0.064, 0.6664	-1.0138	X	0.05291	2s
conf_10	0.1139, -0.6352	-0.9836	X	0.0810	4s
conf_11	0.0000, -0.6668	-0.9878	X	0.1008	5s
conf_12	0.1008, 0.7197	-0.999	X	0.0897	10s
conf_13	0.0736, -0.7112	-1.0307	X	0.0162	2s
conf_14	-0.0898, 0.7126	-1.0316	O	0.0000	4s
conf_15	-0.0890, 0.7113	-1.0317	X	0.0015	2s
conf_16	-0.0898, 0.7126	-1.0316	O	0.0000	4s
conf_17	-0.0890, 0.7113	-1.0317	X	0.0015	5s
conf_18	-0.0909, 0.7134	-1.0317	X	0.0013	10s
conf_19	-0.1014, 0.7143	-1.0312	X	0.0117	2s
conf_20	-0.1041, 0.7219	-1.0303	X	0.0170	4s
conf_21	-0.0898, 0.7126	-1.0316	O	0.0000	2s
conf_22	-0.0903, 0.7136	-1.0317	X	0.0011	4s
conf_23	-0.0898, 0.7126	-1.0316	O	0.0000	5s
conf_24	-0.0898, 0.7126	-1.0316	O	0.0000	10s

A Tabela 6 mostra os resultados obtidos pela execução das configurações descritas na Quadro 4 para o AG ($\mu + \lambda$). Como pode ser visto na Tabela 6, apenas as configurações 23 e 24 encontraram ótimo global. Entretanto, diversas configurações chegaram bem próximas de encontrar este valor ótimo. Se a precisão adotada (de quatro

casas decimais) fosse reduzida, outras configurações seriam capazes de encontrar este valor ótimo.

Tabela 6 - Resultados de otimização da função Sixhump-Camelback para o AG (μ+λ).

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima 1	-0.0898, 0.7126	-1.0316	O	0.0000	xx
Ótima 2	0.0898, -0.7126	-1.0316	O	0.0000	xx
conf_1	0.0000, 0.7201	-0.9987	X	0.0901	2s
conf_2	-0.0525, 0.6876	-1.0222	X	0.0449	4s
conf_3	0.0770, -0.716	-1.0309	X	0.01324	4s
conf_4	-0.1287, 0.7749	-0.9937	X	0.0734	4s
conf_5	0.0700, -0.6979	-1.0287	X	0.0246	5s
conf_6	0.1263, -0.7241	-1.0259	X	0.0382	10s
conf_7	0.0000, -0.6668	-0.9878	X	0.1008	2s
conf_8	0.0416, -0.7085	-1.0226	X	0.0483	4s
conf_9	0.0000, -0.6668	-0.9878	X	0.1008	2s
conf_10	0.2165, -0.68	-0.9587	X	0.1308	4s
conf_11	-0.1451, 0.7129	-1.0199	X	0.0553	5s
conf_12	-0.1147, 0.7203	-1.0290	X	0.0260	10s
conf_13	0.0975, -0.7085	-1.0313	X	0.0087	2s
conf_14	0.0924, -0.7126	-1.0317	X	0.0025	4s
conf_15	-0.0887, 0.7122	-1.0317	X	0.0011	2s
conf_16	0.0880, -0.7112	-1.0317	X	0.0022	4s
conf_17	0.0903, -0.7121	-1.0317	X	0.0007	5s
conf_18	0.0904, -0.7120	-1.0317	X	0.0008	10s
conf_19	0.0747, -0.7050	-1.0304	X	0.0169	2s
conf_20	-0.0910, 0.7137	-1.0317	X	0.0016	4s
conf_21	0.0881, -0.7132	-1.0317	X	0.0018	2s
conf_22	0.0903, -0.7113	-1.0317	X	0.0013	4s
conf_23	-0.0898, 0.7126	-1.0316	O	0.0000	5s
conf_24	-0.0898, 0.7126	-1.0316	O	0.0000	10s

4.1.4 A Função Rastringin

A função de Rastrigin (Törn & Zilinskas, 1989) é uma função altamente multimodal e a localização dos mínimos está distribuída regularmente sendo assim de difícil solução, como pode ser visto na Figura 29. A complexidade do processo de otimização pode ser visualizada na função de Rastrigin pela existência de diversos mínimos locais, que representam soluções não-ótimas e acabam por dificultar o processo de busca pela solução encontrada pelo valor mínimo global.

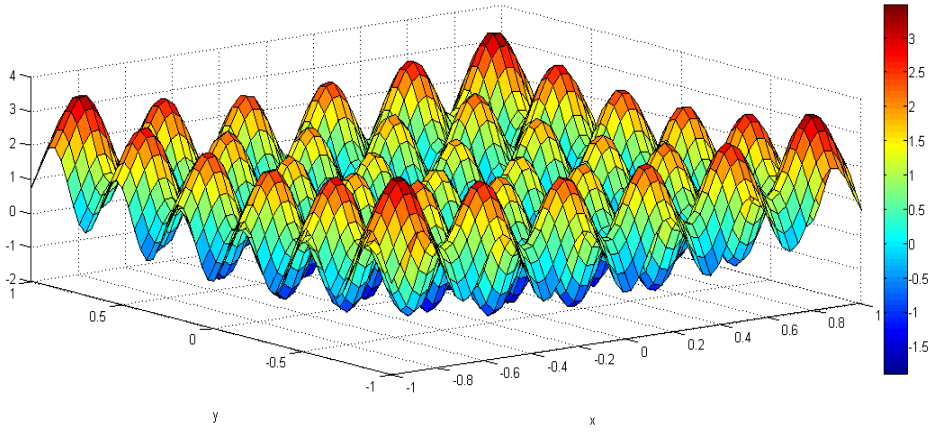


Figura 29 - A função de Rastrigin.

A Tabela 7 mostra os resultados obtidos pela execução das configurações descritas na Quadro 4 para o AG clássico para a otimização da função de Rastrigin. Como pode ser visto na Tabela 7, várias configurações foram capazes de encontrar o mínimo global (configurações 7, 8, 9, 11, 12, 13, 14, 17, 19, 21, 23 e 24). Outras configurações conseguiram obter resultados bastante próximos deste valor ótimo, sendo este valor alcançado caso a precisão numérica fosse reduzida.

Tabela 7 -Resultados de otimização da função Rastrigin para o AG clássico.

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima	0.0000, 0.0000	0.0000	O	0.0000	xx
conf_1	0.0060, -0.0023	-1.9933	X	0.0064	2s
conf_2	0.0112, -0.0122	-1.9555	X	0.0165	3s
conf_3	-0.022, 0.3428	-1.7983	X	0.3435	4s
conf_4	0.0219, 0.011	-1.9032	X	0.0245	4s
conf_5	-0.0037, -0.0022	-1.9970	X	0.0043	4s
conf_6	-0.0040, 0.0000	-1.9974	X	0.0040	8s
conf_7	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_8	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_9	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_10	0.0061, -0.0147	-1.959	X	0.01591	4s
conf_11	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_12	0.0000, 0.0000	0.0000	O	0.0000	8s
conf_13	0.0000, 0.0000	0.0000	O	0.0000	8s
conf_14	0.0000, 0.0000	0.0000	O	0.0000	8s
conf_15	-0.0013, -0.0015	-1.9994	X	0.0019	2s
conf_16	0.0020, 0.0000	-1.9994	X	0.0020	4s
conf_17	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_18	0.0, -0.0010	-1.9999	X	0.0010	8s
conf_19	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_20	-0.0013, 0.0000	-1.9998	X	0.0013	3s
conf_21	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_22	0.0000, -0.0039	-1.9976	X	0.0039	4s
conf_23	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_24	0.0000, 0.0000	0.0000	O	0.0000	8s

A Tabela 8 mostra os resultados obtidos pela execução das configurações descritas na Quadro 4 para o AG (μ + λ). Da mesma forma que no experimento anterior com o AG clássico, o AG (μ+λ) foi capaz de achar o ótimo para várias configurações (configurações 7, 8, 9, 11, 12, 14, 17, 19, 21, 22, 23 e 24).

Tabela 8 - Resultados de otimização da função Rastrigin para o AG (μ+λ).

Configuração	Melhores valores f(x,y)	Valor ótimo encontrado	Precisão	Distância	Tempo de processamento
Ótima	0.0000, 0.0000	0.0000	O	0.0000	xx
conf_1	-0.0072, -0.0088	-1.9790	0.0113	X	2s
conf_2	0.0144, 0.0081	-1.9558	0.0165	X	4s
conf_3	0.3602, -0.0114	-1.8292	0.3603	X	4s
conf_4	-0.0128, -0.0116	-1.9516	0.0172	X	4s
conf_5	-0.0023, -0.0078	-1.9893	0.0081	X	4s
conf_6	0.0060, 0.0075	-1.985	0.0096	X	8s
conf_7	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_8	0.0000, 0.0000	0.0000	O	0.0000	3s
conf_9	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_10	-0.3565, 0.3400	-1.7351	X	0.4926	3s
conf_11	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_12	0.0000, 0.0000	0.0000	O	0.0000	8s
conf_13	-0.0014, 0.0000	-1.9996	X	0.0016	2s
conf_14	0.0000, 0.0000	0.0000	O	0.0000	3s
conf_15	0.0014, 0.0000	-1.9997	X	0.0014	2s
conf_16	-0.0022, 0.0012	-1.9990	X	0.0025	3s
conf_17	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_18	-0.0024, 0.0013	-1.9988	X	0.0027	8s
conf_19	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_20	-0.0023, 0.0	-1.9992	X	0.0023	3s
conf_21	0.0000, 0.0000	0.0000	O	0.0000	2s
conf_22	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_23	0.0000, 0.0000	0.0000	O	0.0000	4s
conf_24	0.0000, 0.0000	0.0000	O	0.0000	8s

Neste primeiro experimento, a ferramenta X-GAT foi aplicada em funções matemáticas que possuem o valor ótimo conhecido. Foram testados os dois AGs implementados, o AG clássico e o AG (μ+λ), com 24 configurações distintas para 4 funções matemáticas. Os resultados mostram que algumas configurações foram capazes de encontrar o valor ótimo do problema. Desta forma, a ferramenta X-GAT pode ser

capaz de encontrar ótimos locais de outros problemas, podendo estes serem ou não os ótimos globais.

4.2 Otimização de Parâmetros de Modelos Hidrológicos

Nesta seção, será mostrada a otimização de dois modelos hidrológicos de transformação de chuva em vazão. São dois os objetivos deste experimento: (1) aplicar o X-GAT em um problema com muitos parâmetros de entrada, ou seja, com muitos genes compondo um cromossomo e verificar se ela foi capaz de obter um bom resultado; e (2) aplicar o X-GAT em um problema que necessite invocar um programa externo implementado em uma linguagem de programação diferente da qual ele foi implementado, gerando assim no final um sistema heterogêneo e avaliar a qualidade da solução obtida.

Primeiramente, é mostrada a calibração (que neste contexto é um sinônimo para otimização) dos parâmetros de entrada do modelo hidrológico *tank model* (Sugawara, 1979), sendo exibidos e analisados seus resultados (Seção 4.2.1). Da mesma forma, é mostrada na seqüência, a calibração do modelo hidrológico WESP (*Watershed Erosion Simulation Program*) (Lopes, 1987), que diferente do primeiro, é um modelo mais complexo e não possui código fonte implementado na linguagem de programação Java, que é a linguagem de programação na qual a ferramenta X-GAT foi implementada (Seção 4.2.2). Dessa forma, para que o X-GAT seja utilizado, é necessária a criação de um sistema heterogêneo (i.e que possa ser executado com códigos fonte implementados em mais de uma linguagem de programação), para que o processo de calibração seja realizado.

O processo cíclico envolvido na transferência do conteúdo úmido do mar para a terra é conhecido como ciclo hidrológico. Os modelos chuva-vazão são usados freqüentemente por profissionais da área de recursos hídricos para gerar séries sintéticas de vazão em pontos de uma bacia hidrográfica. Esses modelos são usados também para estender séries observadas, comumente interrompidas ou muito pequenas para permitir estudos estatísticos.

Uma das maiores dificuldades encontradas no uso desses modelos matemáticos chuva-vazão é a estimativa dos valores de seus parâmetros, que deverão ajustar, adequadamente, as vazões calculadas pelo modelo àquelas observadas em uma bacia hidrográfica (Diniz, 1999). Assim, um processo de calibração adequado é importante para identificar se o modelo é válido para a bacia hidrográfica que está sendo estudada,

como também para gerar informações que permitam tomadas de decisão coerentes com os dados gerados por estes modelos hidrológicos.

A calibração manual (tentativa e erro) é, em geral, extremamente trabalhosa, e exige um grande conhecimento do modelo aplicado e da região hidrológica em estudo, o que é dificultado na aplicação de modelos mais complexos, nos quais existe um grande número de parâmetros a serem estimados. A calibragem automática, além de reduzir estes problemas, introduz o conforto e menos subjetividade nessa fase da modelagem paramétrica (Diniz, 1999).

Dessa forma, torna-se necessário o uso de uma heurística de busca de ótimos globais para realizar o processo de calibração automática. Com o objetivo de mostrar as facilidades inseridas pela utilização da ferramenta X-GAT, são mostrados na seqüência, os processos de calibração e validação dos modelos hidrológicos *tank model* e do WESP.

4.2.1 Otimização do modelo hidrológico *tank model*

O *tank model* pode ser chamado de um sistema de vazão com armazenamento, composto por uma série de tanques. A bacia hidrográfica é representada por um conjunto de tanques, e a saída ou escoamento de cada tanque é assumido como sendo proporcional à altura da água a partir da posição da descarga ou infiltração. A profundidade do tanque é assumida como sendo o armazenamento na bacia hidrográfica.

Para esta simulação, o *tank model* foi implementado com três tanques, e a Figura 30 mostra como ele foi implementado. A variável P representa a precipitação (chuva) que ocorreu em um determinado tempo *t*. São também mostradas as variáveis de vazão (a1, a2, a3, a4, y1, y2, y3 e y4), as infiltrações (b1, b2, b3, z1, z2 e z3), as alturas de escoamento (h1, h2, h3 e h4) e a profundidade de armazenamento (X1, X2 e X3).

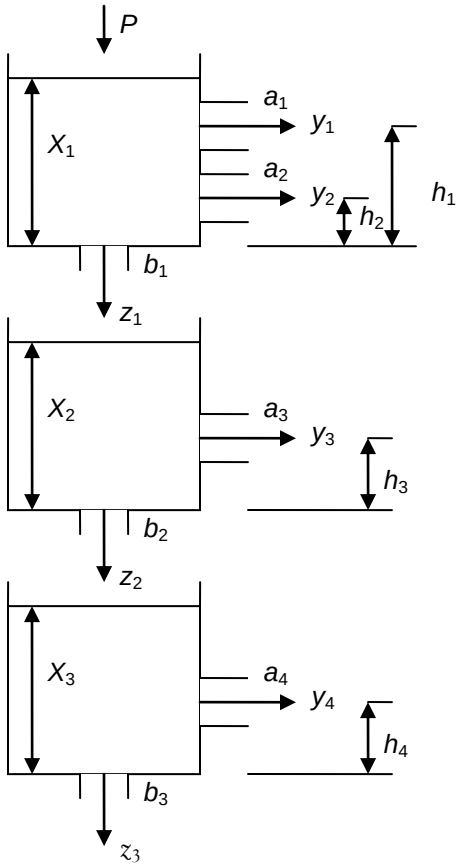


Figura 30 - O modelo *tank model* com três tanques.

Os parâmetros a serem otimizados são as vazões a1, a2, a3 e a4; as infiltrações b1, b2 e b3; e a altura do escoamento h1, h2, h3 e h4. Esses valores e o *tank model* são definidos pelas expressões mostradas na Equação 6.

Equação 6 - Equações do modelo hidrológico *tank model*.

$$y_1(t) = a_1[X_1(t) - h_1] \quad (1)$$

$$y_2(t) = a_2[X_1(t) - h_2] \quad (2)$$

$$y_3(t) = a_3[X_2(t) - h_3] \quad (3)$$

$$y_4(t) = a_4[X_3(t) - h_4] \quad (4)$$

$$z_1(t) = b_1 X_1(t) \quad (5)$$

$$z_2(t) = b_2 X_2(t) \quad (6)$$

$$z_3(t) = b_3 X_3(t) \quad (7)$$

$$Q(t) = y_1(t) + y_2(t) + y_3(t) + y_4(t) \quad (8)$$

$$X_1(t) = X_1(t-1) + P(t) - y_1(t) - y_2(t) - z_1(t) \quad (9)$$

$$X_2(t) = X_2(t-1) + z_1(t) - y_3(t) - z_2(t) \quad (10)$$

$$X_3(t) = X_3(t-1) + z_2(t) - y_4(t) - z_3(t) \quad (11)$$

onde t é dado em dias; y1(t), y2(t), y3(t) e y4(t) são as vazões estabelecidas no dia t; z1(t), z2(t) e z3(t) são os valores de infiltração em cada tanque no dia t; X1(t), X2(t) e X3(t) são a profundidade de armazenamento no dia t; Q(t) é o total da vazão no dia t; e P(t) é a precipitação no dia t.

O processo de calibração consiste em achar um conjunto de valores para os parâmetros de forma que estes valores possam atingir o ótimo global. Os parâmetros a serem otimizados nesse problema são a1, a2, a3, a4, b1, b2, b3, h1, h2, h3 e h4. A faixa de variação dos valores que essas variáveis podem assumir é mostrada na Quadro 5.

Quadro 5 - Faixa de variação dos parâmetros de entrada a serem otimizados na simulação.

Parâmetros	a_1	a_2	a_3	a_4	b_1	b_2	b_3	h_1	h_2	h_3	h_4
Limite inferior	0.001	0.001	0.01	0.01	0.15	0.01	0.0	10	10	10	10
Limite superior	0.1	0.1	0.3	0.7	0.9	0.04	0.3	90	120	120	120

Com o objetivo de executar a otimização do modelo, foi necessária a seleção de uma função de aptidão, que para o presente estudo, foi escolhida a função descrita pela Equação 7.

Equação 7 – Função de aptidão adotada para otimização do *tank model*.

$$Minimizar \sum_{t=1}^{N} \frac{|Q_O - Q_C|}{Q_O}$$

onde Qo é a vazão observada, Qc é a vazão calculada pelo modelo hidrológico, e N é o número de dias utilizado na calibração dos parâmetros para o modelo.

O procedimento de calibração dos parâmetros de entrada do modelo hidrológico *tank model* é o mesmo adotado por (Santos *et al*., 2003). Para a calibração dos parâmetros de entrada de um modelo hidrológico, é usado certo período de dados. Os parâmetros de entrada são ajustados automaticamente utilizando-se uma heurística de busca para ótimos globais, e é nessa calibração automática em que o X-GAT é empregado. A partir da extensão da classe *PopulationEvaluationFactory*, mostrada na Seção 3.4.5, a função de aptidão mostrada na Equação 7 é implementada e a execução do modelo hidrológico é realizada com os genes gerados pelo X-GAT.

Nessa simulação, os dados de vazão do reservatório do rio Ishite foram utilizados. Esse reservatório abastece a cidade de Matsuyama no Japão. O período utilizado para a calibração dos parâmetros de entrada foram os dados dos anos de 1992 a 1993, enquanto os anos de 1994 a 1995 foram utilizados para a validação do *tank model* calibrado.

Para avaliar os resultados obtidos, os índices estatísticos de correlação (r) e erro do tipo *bias* (B) foram utilizados como critério para avaliação do desempenho da calibração do *tank model*. A correlação calcula a variabilidade de um número de previsões ao redor do valor verdadeiro. O erro do tipo *bias*, por outro lado, é a medida do erro sistemático e, assim, calcula o grau com que a previsão está consistentemente acima ou abaixo do valor real. Obter apenas uma alta correlação não significa necessariamente obter uma alta precisão. Por exemplo, um significante constante erro do tipo *bias* nas previsões forneceria uma correlação alta (r = 1), mas de fraca precisão. Como resultado, a precisão de previsões é melhor analisada pelo uso de ambas. Uma previsão perfeita, o

que é praticamente impossível de acontecer, deve possuir r = 1 e B = 0. Em (Salas, 1993), o autor fornece as equações para o cálculo desses índices.

A Figura 31 e a Figura 32 apresentam uma comparação entre as vazões observadas e calculadas para a calibração e validação dos dados, respectivamente. Esses gráficos mostram que o *tank model* calibrado foi bem eficiente para estimar as vazões de entrada no reservatório. As altas correlações e valores baixos para o erro *bias* entre os valores observados e calculados indicam que X-GAT foi capaz de calibrar o *tank model* eficientemente e, podendo assim ser bastante útil para a calibração de outros modelos hidrológicos aplicados à área de recursos hídricos.

Figura 31 - Comparação entre vazões observadas e calculadas para calibração do modelo hidrológico *tank model*.

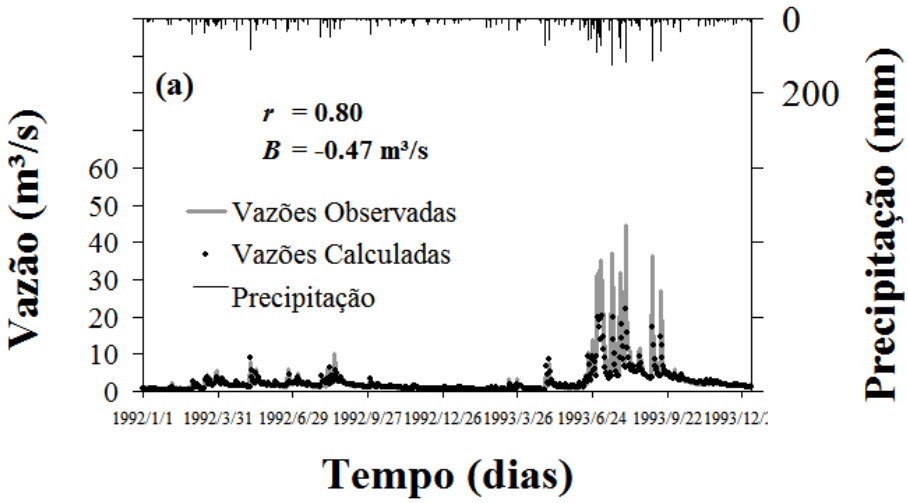
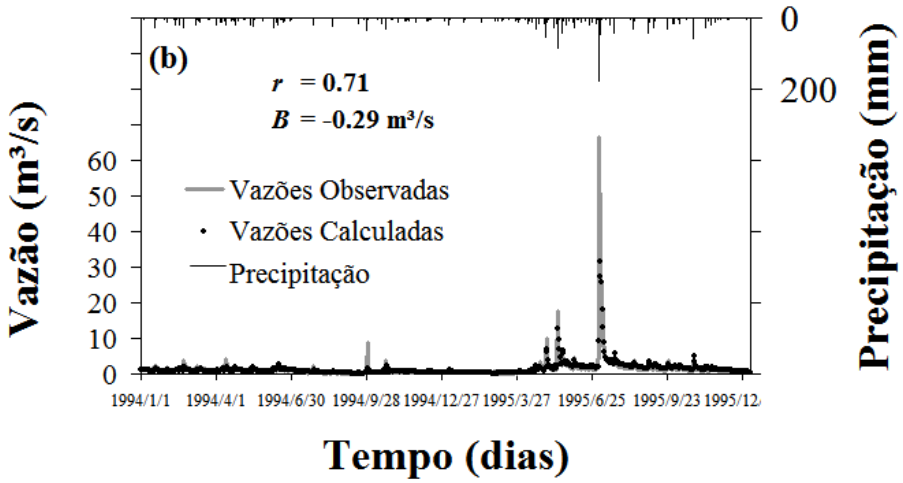


Figura 32 - Comparação entre vazões observadas e calculadas para validação do modelo hidrológico *tank model*.



4.2.2 Otimização do modelo hidrológico WESP

Lopes & Lane (1988) desenvolveram um modelo hidrossedimentológico de base física chamado WESP que calcula o escoamento e a produção de sedimentos baseado na aproximação de ondas cinemáticas para o escoamento superficial devido ao excesso de chuva *re* (m/s), que é obtido pela subtração da taxa de infiltração f(*t*) da intensidade da chuva I, (i.e. *re* = I – f(t)). O modelo foi desenvolvido para pequenas bacias para gerar o hidrograma (diagrama hídrico) e o respectivo sedigrama (diagrama de sedimentos).

Equação 8 - Equações do modelo WESP.

$$f(t) = K_s\left(1 + \frac{\Delta\theta\psi}{F(t)}\right) \quad (1)$$

$$N_s = \Delta\theta\psi_i = (\theta_s - \theta_i)\psi_i \quad (2)$$

$$u = \frac{1}{n} R_H^{2/3} S_f^{1/2} \quad (3)$$

$$u = \alpha h^{m-1} \quad (4)$$

$$\frac{\partial h}{\partial t} + \alpha m h^{m-1} \frac{\partial h}{\partial x} = r_e \quad (5)$$

$$\frac{\partial(ch)}{\partial t} + \frac{\partial(cuh)}{\partial x} = e_I + e_R - d \quad (6)$$

$$e_I = K_I I r_e \quad (7)$$

$$e_R = K_R \tau^{1.5} \quad (8)$$

$$d = \varepsilon_p V_s c \quad (9)$$

$$V_s = F_o \sqrt{\frac{(\gamma_s - \gamma)}{\gamma} g d_s} \quad (10)$$

$$F_o = \sqrt{\frac{2}{3} + \frac{36\nu^2}{g d_s^3\left(\frac{\gamma_s}{\gamma} - 1\right)}} - \sqrt{\frac{36\nu^2}{g d_s^3\left(\frac{\gamma_s}{\gamma} - 1\right)}} \quad (11)$$

$$Q = \alpha A R_H^{m-1} \quad (12)$$

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = q_A \quad (13)$$

$$\frac{\partial AC}{\partial t} + \frac{\partial CQ}{\partial x} = q_s + e_r - d_c \quad (14)$$

$$e_r = a(\tau - \tau_c)^{1.5} \quad (15)$$

$$d_c = \varepsilon_c T_W V_s C \quad (16)$$

O processo de infiltração é modelado com a equação de Green-Ampt (Green & Ampt, 1911), que pode ser escrita na forma mostrada na Equação 8(1), onde K_s é a condutividade hidráulica do solo saturado (m/s), $F(t)$ é a altura acumulada da água infiltrada (m), ψ é a altura de sucção média na frente de molhamento (m), $\Delta\theta$ é a variação do conteúdo úmido, e t é a variável tempo (s).

O conteúdo úmido θ e a altura de sucção ψ podem se expressas como um único parâmetro que poderia ser chamado de parâmetro umidade-tensão do solo N_s, mostrado na Equação 8(2), onde θ_s é o conteúdo úmido na saturação, que é quase igual à porosidade e θ_i é o conteúdo úmido inicial do solo. O escoamento superficial é considerado como escoamento laminar superficial ou escoamento concentrado nos canais.

O escoamento laminar superficial e espacialmente variado é considerado unidimensional, descrito pela equação do escoamento turbulento de Manning, que pode ser vista na Equação 8(3): onde u é a velocidade média local do escoamento (m/s), $R_H(x,t)$ é o raio hidráulico (m), S_f é a declividade de atrito e n é o fator de atrito de Manning. Assim, a velocidade local para o escoamento nos planos pode ser obtida considerando o raio hidráulico igual à altura do escoamento ($R_H = h$) e usando a aproximação de ondas cinemáticas, obtém-se que a declividade de atrito é igual a declividade do plano ($S_0 = S_f$), como mostra a Equação 8(4): onde h é a altura do escoamento (m), α é um parâmetro relacionado a declividade e rugosidade da superfície, igual a $(1/n)S_0^{1/2}$, e m é um parâmetro de geometria cujo valor é igual a 5/3 para seções retangulares largas.

A equação de continuidade para o plano unidimensional corresponde então, à Equação 8 (5). A partir da Equação 8(4) e da Equação 8(5), a velocidade e altura do escoamento superficial (u, h) podem ser calculadas para um dado excesso de chuva r_e. O transporte de sedimentos é considerado como a taxa de erosão no plano menos a taxa de deposição no trecho. A erosão ocorre devido ao impacto das gotas de chuva bem como à tensão de cisalhamento do escoamento. Assim, a equação de continuidade para o transporte de sedimentos é expressa pela Equação 8(6): onde c é a concentração de sedimentos no escoamento superficial (kg/m^3), e_I é a taxa de erosão de sedimentos devido ao impacto da chuva (kg s/m^2), e_R é a taxa de erosão devido à tensão de cisalhamento do escoamento (kg s/m^2), e d é a taxa de deposição de sedimentos (kg

s/m²). A taxa de erosão devido ao impacto da gota de chuva e_I é uma função da taxa de desprendimento pelo impacto das gotas de chuva e a taxa de transporte de partículas pelo escoamento raso, expressa pela Equação 8(7), onde K_I é o parâmetro de desprendimento do solo (kg s/m⁴). A taxa de erosão de sedimentos devido à tensão de cisalhamento e_R é expressa (Croley, 1982; Foster, 1982) pela Equação 8(8); onde K_R é um fator de erodibilidade do cisalhamento (kg m/N$^{1.5}$ s), e τ é tensão cisalhante efetiva (N/m²), que é dada por $\tau = \gamma h S_f$, sendo γ o peso específico da água (N/m³).

A taxa de deposição de sedimentos d na Equação 8(6) é dada (Einstein, 1968) pela Equação 8(9), onde ε_p é um coeficiente que depende das propriedades do fluido e sedimentos, igual a 0,5 no presente estudo por se baseado em Davis (1978), $c(x,t)$ é a concentração de sedimentos em transporte (kg/m³), e V_s é a velocidade de queda das partículas (m/s) calculado pela equação de Rubey (Equação 8 (10) e Equação 8 (11)), onde γ_s é o peso específico do sedimento (N/m³), ν é a viscosidade cinemática da água (m²/s), d_s é o diâmetro médio dos sedimentos (m), e g é a aceleração da gravidade (m/s²).

O escoamento concentrado nos canais é também descrito pelas equações de continuidade e do momento. A equação do momento pode ser reduzida para equação do escoamento com a aproximação de ondas cinemáticas 8(12): onde Q é a vazão (m³/s), e A é a área da seção transversal do escoamento (m²). A equação de continuidade para o escoamento no canal é dada pela Equação 8 (13): onde q_A é a entrada lateral de escoamento por unidade de comprimento do canal. A Equação 8(12) e a Equação 8(13) permitem o cálculo do escoamento no canal. Como o efeito do impacto das gotas de chuva é desprezível no canal, a equação de continuidade para os sedimentos é expressa sem o componente do impacto da chuva, representado pela Equação 8 (14), onde $C(x,t)$ é a concentração de sedimentos em transporte no canal (kg/m²), q_s é a entrada lateral de sedimento no canal (kg s/m), d_c é a taxa de deposição de sedimentos no canal (kg s/m), e e_r é a taxa de erosão do material do leito do canal (kg s/m). Os componentes do fluxo de sedimentos para os canais são dados da seguinte forma: a taxa de erosão do material do leito do canal e_r é obtido de uma equação geral, inicialmente desenvolvida para a capacidade de transporte da carga de leito (Croley, 1982; Foster, 1982), representada pela Equação 8(15), onde $\tau_c = \delta(\gamma_s - \gamma)d_s$ a é o parâmetro de erodibilidade dos sedimentos, e τ_c é tensão de cisalhamento crítica (N/m²), que é dada por, onde δ é um coeficiente, igual a 0,047 no presente estudo, γ_s é o peso específico dos sedimentos

(N/m³), e d_s é o diâmetro médio dos sedimentos (m). A taxa de deposição de sedimentos no canal d_c (kg s/m) na Equação 8(14) é expressa pela (Mehta, 1983) Equação 8(16): onde ε_c é o parâmetro de deposição para os canais, considerado igual a 1,0 no presente caso, e T_W é a largura superior do canal (m). A partir da Equação 8(14), a taxa de transporte de sedimentos (CQ) pode se calculada para o escoamento concentrado nos canais com A e Q obtidos a partir da Equação 8 (13).

Como área de estudo, foram utilizados dados das microbacias da bacia experimental de Sumé, localizada na região nordeste do Brasil. Essa bacia experimental foi operacionalizada de 1982 até 1991 pela SUDENE (Superintendência do Desenvolvimento do Nordeste), ORSTOM (*French Office of Scientific Research and Technology for Overseas Development*) e UFPB (Universidade Federal da Paraíba) para obter dados de campo para o cálculo de transporte de sedimentos produzidos pela chuva no ambiente de estudo (Cadier *et al.*, 1983).

Utilizando o trabalho de Santos et al. (1994), foram selecionados 21 eventos que aconteceram no ano de 1988 e 25 eventos que aconteceram entre 1988 e 1991, dando um total de 46 eventos. Esses períodos foram escolhidos porque a bacia foi administrada nesse período sobre boas condições de operacionalização, dando assim confiabilidade aos dados colhidos durante este período.

Como função de aptidão utilizada no processo de calibração automática, foi adotada a Equação 9.

Equação 9 - Função de aptidão usada na otimização do modelo WESP.

$$\textbf{Minimizar } J = (E_o - E_c)^2 + (L_o - L_c)^2,$$

onde E_o e E_c são respectivamente os sedimentos observados e calculados, e L_o e L_c são respectivamente as vazões observadas e calculadas.

Existem duas diferenças importantes entre este experimento e o experimento anterior com o modelo hidrológico *tank model* (Seção 4.2.1). A primeira diz respeito ao resultado de saída do modelo hidrológico. No *tank model*, são calculadas apenas as vazões de saída, enquanto que neste experimento, são calculadas a vazão e o transporte de sedimentos. A segunda diz respeito ao código fonte dos modelos hidrológicos, já que o *tank model* foi implementado na linguagem de programação Java, que é a mesma da ferramenta X-GAT. Já o modelo WESP foi implementado na linguagem de programação

Fortran. Dentro da nova classe criada para avaliar os indivíduos, estendida da classe abstrata *PopulationEvaluationMethod*, é feita uma chamada externa a um programa executável em Fortran. Este programa, quando invocado, faz a leitura dos valores dos parâmetros de entrada que estão em um arquivo com o formato .txt. Uma vez que os valores dos parâmetros de entrada são gerados pelo X-GAT, o AG durante sua execução manipula este arquivo .txt de entrada, escrevendo os valores de entrada gerados pela heurística. Uma vez executado o modelo hidrológico WESP, os valores de E_c e de L_c são lidos pelo X-GAT e utilizados para computar a função de aptidão. Cada novo indivíduo gerado pelo X-GAT executa este processo, até que o AG convirja para um determinado valor. Dessa forma não houve necessidade de re-implementação do modelo WESP, tornando assim desnecessário qualquer processo de validação da implementação do modelo hidrossedimentologico, o que seria bastante custoso.

O X-GAT foi usado para a otimização dos quatro parâmetros do modelo hidrossedimentológico. O primeiro foi o parâmetro de tensão-umidade do solo N_s, presente na Equação 8(2) e está relacionado somente ao escoamento, como depende a umidade antecedente do solo, este parâmetro deve ser diferente para cada evento e sua média não pode ser calculada. Os outros três parâmetros estão relacionados à erosão, seja ela nos planos ou nos canais, são eles: o parâmetro de erosão devido à força cisalhante do escoamento no canal a, presente na Equação 8(15), e o mesmo para os planos K_R, Equação 8(8), e o parâmetro de entrada de sedimentos devido aos impactos das gotas de chuva K_I, Equação 8(7). Os valores médios encontrados foram a = 0.01 kg m^2, K_R = 3.12 kg m/N$^{1.5}$ s, e K_I = 5.17 $\times$ 10^8 kg s/m^4.

A Figura 33 e a Figura 34 mostram as comparações entre as vazões calculadas e as vazões observadas e os sedimentos calculados e os sedimentos observados, respectivamente. Como pode ser observado, o X-GAT conseguiu obter resultados satisfatórios no processo de calibração automática destes eventos, pois com exceção de alguns eventos, os resultados obtidos pelos parâmetros médios encontrados se aproximaram bastante do valor observado.

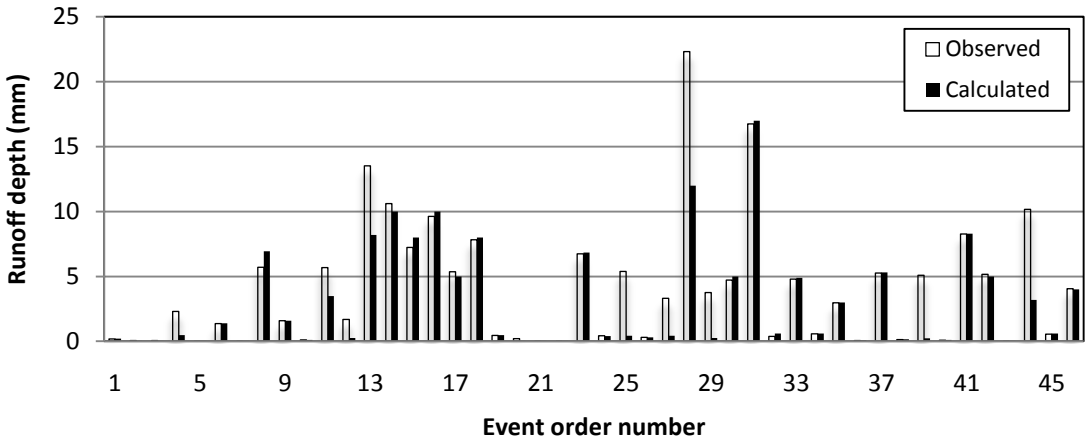


Figura 33 - Comparação entre as vazões calculadas e observadas para o modelo WESP.

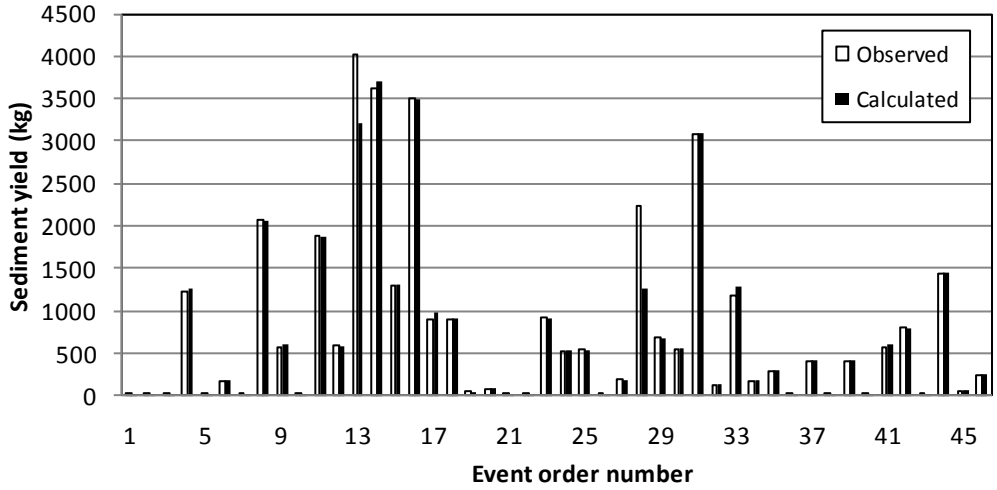


Figura 34 - Comparação entre os sedimentos calculados e observados para o modelo WESP.

Nesta seção, foram mostrados dois processos de otimização de parâmetros de entrada de modelos hidrológicos de transformação de chuva em vazão. Os dois objetivos deste experimento foram alcançados. Para o modelo hidrológico *tank model*, que possui 11 parâmetros de entrada a serem otimizados, o experimento foi realizado com sucesso e bons resultados foram obtidos, já que o valor correlação próximo ao valor 1.0 (i.e 0.71) e o valor de bias próximo a 0.0 (i.e -0.29) foram obtidos. Para o experimento com o modelo WESP, a invocação do código deste modelo que roda na linguagem de programação diferente da usada pela ferramenta de otimização, gerou um sistema heterogêneo, que roda parte de sua execução em Java (utilizada na implementação do X-

GAT) e parte na linguagem de programação Fortran (utilizada na implementação do modelo WESP). Além disso, graficamente (mostrados na Figura 33 e na Figura 34), nota-se que os parâmetros de entrada gerados pela ferramenta X-GAT quase sempre foram capazes de encontrar o valor exato, com exceção de alguns poucos onde o valor encontrado não foi capaz de encontrar valores próximos de vazão e de transporte de sedimentos.

4.3 Otimização de Parâmetros de Algoritmos de Clusterização de Objetos Móveis

Nesta seção, será mostrada a otimização dos parâmetros de entrada de dois algoritmos de clusterização de pontos de trajetória de objetos móveis: o CB-SMoT (*Clustering-based Stops and Moves Trajectories*) (Palma et al., 2008) que clusteriza (i.e. agrupa pontos da trajetória em *stops*) trajetórias a partir da variação da velocidade média do objeto móvel; e o DB-SMoT (*Direction-based Stops and Moves Trajectories*) (Manso et al., 2010), que clusteriza trajetórias a partir da variação da direção do objeto móvel. O objetivo deste experimento é utilizar a ferramenta X-GAT em mais um domínio de aplicação, mostrando assim que esta ferramenta pode ser aplicável em qualquer área da ciência onde a heurística de busca com AGs possa ser utilizada.

Antes de definir detalhadamente o experimento realizado, é feita uma contextualização desta área na qual o X-GAT foi aplicado (Seção 4.3.1). Na seqüência, é mostrado o processo de análise ROCCH (*Receiver Operating Characteristic Convex Hull*), que é a metodologia utilizada no cálculo da função de aptidão do AG utilizado no processo de calibração (Seção 4.3.2). Por último, é caracterizado o experimento realizado e são mostrados os resultados do processo de otimização (Seção 0).

4.3.1 Trajetórias de objetos móveis

Os recentes avanços na tecnologia de dispositivos móveis como aparelhos de GPS, telefone celular e redes de sensores têm facilitado a coleta de grandes volumes de dados espaço-temporais. Dados brutos deste tipo geralmente escondem conhecimentos e informações que podem ser interessantes e úteis em processos de tomada de decisão nas mais diversas áreas de aplicação. A extração de conhecimento a partir desse tipo de dado envolve a aplicação de conceitos e tarefas de mineração de dados para o enriquecimento de trajetórias com informações semânticas. Uma trajetória bruta pode ser enriquecida com diferentes informações semânticas, as quais variam de acordo com o contexto da

aplicação e o problema que o usuário pretende resolver (Bogorny & Wachowicz, 2008). O contexto no qual a aplicação é construída é fundamental para extração de informações relevantes. Em uma aplicação de trânsito, por exemplo, pode ser interessante que sejam adicionadas às trajetórias, informações de lugares transitados como rotatórias, cruzamentos, sinaleiras, pardais, etc. Já em uma aplicação de turismo, as partes interessantes da trajetória do objeto móvel podem ser pontos turísticos, hotéis, aeroporto, dentre outras. Além disso, os tipos de comportamentos dos objetos móveis analisados a partir dos dados brutos de trajetórias geradas por dispositivos móveis possuem uma relevância diferente dependendo do contexto em que estes são estudados. Para uma trajetória de trânsito, pode ser interessante avaliar as variações de velocidade, por exemplo, para tentar inferir que um veículo tem estado em um engarrafamento em certos períodos de tempo. Entretanto, para avaliar a trajetória de um barco pesqueiro, pode ser mais interessante avaliar sua variação de direção, para detectar os locais onde e quando o barco realiza atividades de pesca.

De acordo com Spaccapietra et al. (2007), uma trajetória é definida como a evolução da situação (entendida como um ponto no espaço) de um objeto que se move no espaço durante um determinado intervalo de tempo e visa atingir um determinado objetivo. Esta definição exibe o conceito de semântica de trajetórias, na qual cada trajetória é vista como um conjunto de *stops* e *moves*. Os *stops* são vistos como a parte mais importante da trajetória, pois do ponto de vista da aplicação, eles são os comportamentos interessantes realizados pelo objeto móvel. Já os *moves* são os movimentos realizados pelos objetos móveis entre os *stops*.

Dessa forma, para uma aplicação de mapeamento de pássaros migratórios, os *stops* podem ser vistos como as regiões das trajetórias em que os pássaros se alimentam ou se reproduzem. Os *moves* podem ser vistos como as rotas por onde os pássaros se locomoveram até a chegada em um local apropriado para se alimentar ou se reproduzir.

Para este experimento são utilizados dois algoritmos de clusterização (CB-SMoT e DB-SMoT) que extraem padrões de comportamento com o objetivo de encontrar *stops* e *moves*. Em Palma et al. (2008), é apresentado o método CB-SMoT, que se baseia na variação da velocidade do objeto móvel como parâmetro de entrada. Este algoritmo faz uma análise seqüencial dos pontos da trajetória, criando clusters dos pontos com variação de velocidade semelhantes, e estes clusters representam os *stops*. Da mesma forma, em Manso et al. (2010), é apresentado o método DB-SMoT. O DB-SMoT gera os

clusters que representam os *stops* candidatos considerando a direção, e não mais a velocidade, como o CB-SMoT.

4.3.2 Análise ROCCH

Antes de definir e mostrar como a análise ROCCH foi utilizada neste experimento, será realizada uma breve contextualização da análise ROC (*Receiver Operating Characteristic*), que faz parte da análise ROCCH. A curva ROC foi desenvolvida no contexto de detecção de sinais eletrônicos e problemas com radares, durante a segunda guerra mundial (Martinez et al., 2003). Seu objetivo era quantificar a habilidade dos operadores dos radares (chamados originalmente de *receiver operators*) em distinguir um sinal de um ruído.

No contexto deste experimento, os algoritmos de clusterização podem ser vistos como modelos de classificação, pois classificam os dados espaço-temporais em *stops* e *moves*. A análise do desempenho de um modelo de classificação pode ser medida pela matriz de confusão, a qual tem duas entradas para as classes desejadas e duas entradas para as classes previstas, conforme mostrado na Quadro 6 (Fawcett, 2006). A matriz de confusão da análise ROC é uma forma de apresentar dados estatísticos de avaliação para um classificador. Um evento é dito como verdadeiro positivo (TP) quando um exemplo positivo é classificado como positivo. Um evento é dito como falso positivo (FP) quando um exemplo negativo é classificado como positivo. Da mesma forma, um evento é dito como verdadeiro negativo (TN) é quando um exemplo negativo é classificado como negativo. Por último, um evento é dito como falso negativo (FN) quando um exemplo positivo é classificado como negativo.

Quadro 6 - Matriz de confusão da análise ROC.

	Predição Positiva	**Predição Negativa**
Classe Positiva	Verdadeiro Positivo – TP	Falso Negativo – FN
Classe Negativa	Falso Positivo – FP	Verdadeiro Negativo – TN

As métricas mais importantes obtidas a partir da análise da matriz de confusão são a precisão, exatidão (do inglês, *accuracy*), sensibilidade (TP rate) e falso alarme (FP rate). Estas métricas podem ser calculadas a partir das equações vistas na Quadro 11.

Quadro 7 - Principais métricas da análise ROC.

Métrica	**Fórmula**
Precisão	TP / (TP + FP)
Exatidão	(TP + TN) / (TP + FP + FN + TN)
TP rate (Sensibilidade)	TP / (TP + FN)
FP rate (Falso Alarme)	FP / (FP + TN)

A partir destas equações é possível gerar o espaço ROC. O espaço ROC mostra a relação das taxas de falsos positivos (FP rate) e verdadeiros positivos (TP rate), sendo assim um espaço bidimensional.

A Figura 35 ilustra o espaço ROC, destacando os pontos e as regiões importantes no processo de análise. O ponto que norteia o processo de classificação tem o par ordenado (0.0, 1.0) como suas coordenadas, e representa o classificador perfeitono qual todas as instâncias positivas e negativas são corretamente classificadas. O ponto de coordenadas (1.0, 0.0), define o modelo que sempre efetuará classificações erradas. A origem do sistema de coordenadas caracteriza o método que nunca classifica uma instância como positivo. O ponto de coordenadas (1.0, 1.0) sempre classifica um exemplo como positivo. A diagonal que une o ponto (0.0, 0.0) ao ponto (1.0, 1.0) representa um classificador de desempenho aleatório.

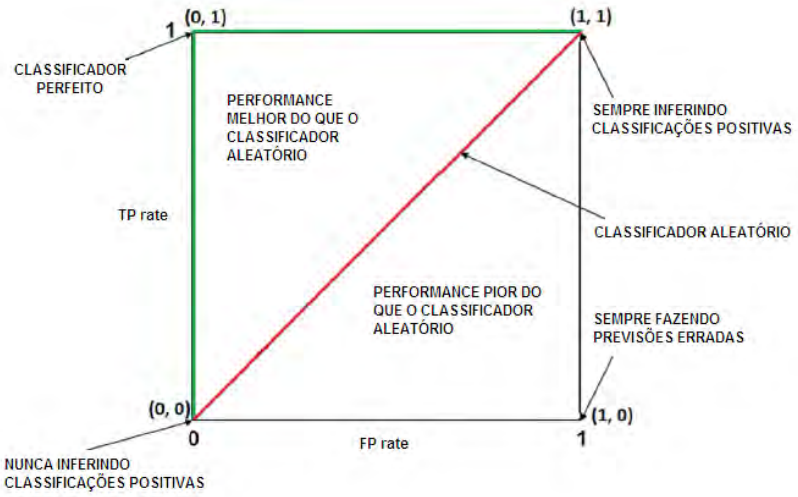


Figura 35 - Espaço ROC, adaptado de Fawcett (2006).

No espaço ROC, a partir do *convex hull* (em português, envelope externo convexo), é possível avaliar o desempenho de um determinado conjunto de métodos de classificação binária. A partir da execução dos classificadores, os seus resultados podem ser projetados como pontos (FP_rate, TP_rate) no espaço ROC. Neste caso, o *convex hull* é um polígono que tem como um lado a linha que compreende o classificador aleatório e os demais lados definidos pela ligação entre todos os pontos que possuem a menor distância para o classificador perfeito (Prati et al. 2008). Os modelos compreendidos no *convex hull* são considerados os de melhor desempenho em função dos custos atribuídos aos resultados da matriz de confusão na aplicação em questão. Os demais pontos não localizados no *convex hull* representam modelos que podem ser descartados, pois têm desempenhos inferiores (Prati et al. 2008).

O processo de análise de desempenho dos classificadores é feito percorrendo-se os pontos do *convex hull* com uma linha tangente denominada de isodesempenho Os pontos que interceptarem a linha de isodesempenho ajustada para o problema em questão representarão os modelos com melhor desempenho. O ajuste desta linha pode ser feito sob dois critérios: custo de classificação ou desproporção das instâncias. No contexto deste experimento, o custo para a classificação de um *stop* como um *move* ou de um *move* como um *stop* é o mesmo. Entretanto, é importante salientar que a distribuição das classes positivas e negativas nos experimentos é desbalanceada, sendo assim utilizada a linha de isodesempenho para fazer o balanceamento das classes.

O ajuste da linha de isodesempenho para balancear as proporções de instâncias positivas e negativas pode ser realizado a partir da Equação 10. Esta equação é utilizada como função de aptidão deste experimento, sendo este valor maximizado para que se obtenha a melhor configuração dos classificadores utilizados.

Equação 10 - Equação para cálculo da linha de isodesempenho.

$$\textbf{isometric line} = TP_{rate} * PP - FP_{rate} * PN$$

onde PP e PN são as proporções de instâncias positivas e negativas, respectivamente.

4.3.3 Otimizando os parâmetros de entrada do CB-SMoT e DB-SMoT

Após uma breve introdução da área de trajetórias de objetos móveis e da análise ROOCH, esta seção caracteriza e avalia o experimento realizado. Como dito anteriormente, os algoritmos de clusterização CB-SMoT e DB-SMoT classificam as partes da trajetória em *stops* e *moves*. Para este experimento, serão utilizados dados de

barcos que realizam atividades de pesca pelágica no litoral nordestino. Os algoritmos de clusterização devem inferir corretamente em quais áreas da trajetória os barcos estavam efetivamente realizando a atividade de pesca. Para este experimento, foram utilizadas duas trajetórias de barcos de mesmo porte.

Primeiramente, iremos caracterizar as duas trajetórias utilizadas no experimento. Então, serão mostrados os parâmetros de entrada e os espaços de busca para cada algoritmo de clusterização, e as configurações utilizadas pelo X-GAT no processo de otimização com seus respectivos resultados.

A primeira trajetória utilizada no processo de calibração é a do barco Chung Kuo 85, que partiu da costa brasileira realizando atividades de pesca com duração de 34 dias. Do tempo total desta trajetória, 23 dias correspondem a operações de pesca e os outros 11 dias correspondem a dias de navegação do porto para área de pesca, de uma área de pesca para outra e, finalmente, da última área de pesca para o porto. Esta trajetória tem um total de 1142 pontos, dos quais 850 pontos abrangem as operações de pesca, que correspondem ao lançamento e recolhimento do espinhel, e 298 pontos estão associados aos 11 dias de navegação, sem atividades de pesca. Sendo assim, esta trajetória tem uma porcentagem de 74% de instâncias positivas e 26% de exemplos negativos. Em média, a cada 42 minutos, um registro espaço-temporal foi gravado, a partir do receptor GPS a bordo da embarcação, gerando a trajetória em questão. Esta trajetória pode ser vista na Figura 36. Os pontos na cor cinza representam os *moves* (i.e. áreas onde o barco estava se deslocando) e os pontos na cor preta representam os *stops* (i.e áreas onde o barco estava realizando atividades de pesca).

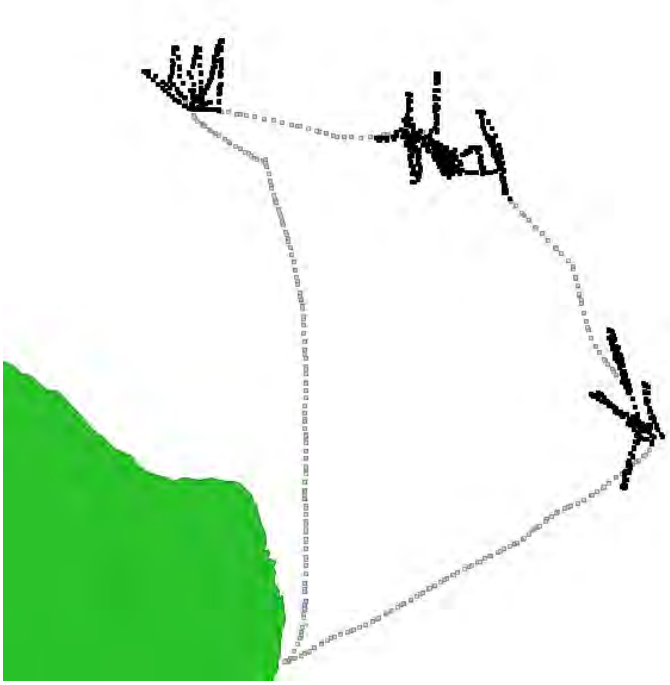


Figura 36- Trajetória do barco Chung Kuo 85.

A segunda trajetória utilizada no processo de calibração é a do barco Chung Kuo 287, que realizou atividades de pesca com duração de 44 dias, retornando ao porto de origem. Do tempo total gasto nesta atividade, 38 dias foram efetivamente utilizados em operações de pesca e os outros 6 dias em navegação do porto para área de pesca e da última área de pesca para porto. Esta trajetória tem um total de 1575 pontos, sendo 1379 pontos abrangendo as operações de lançamento e recolhimento do espinhel, e 196 pontos que abrangem os 6 dias de navegação. Esta trajetória teve uma média de 40 minutos por gravação de cada ponto. Esta trajetória, tendo uma percentagem de 87.5% de instâncias positivas e 12.5% de exemplos negativos, pode ser vista na Figura 37. Os pontos na cor cinza representam os *moves* e os pontos na cor preta representam os *stops*, como mostrado nesta figura.

Figura 37 - Trajetória do barco Chung Kuo 287.

Essas duas trajetórias foram utilizadas no processo de otimização, caracterizando assim o problema de otimização com 2717 pontos de trajetória bruta (incluindo os *stops* e os *moves*). Dessa forma, foram computados 2229 pontos que representam *stops* (proporção de 82% de instâncias positivas) e 448 pontos que representam *moves*.

Apresentadas as trajetórias utilizadas neste experimento, serão definidos os espaços de busca de cada um dos algoritmos de clusterização. O algoritmo de clusterização CB-SMoT possui três parâmetros de entrada: velocidade máxima permitida entre um ponto e outro dentro de um cluster (speedLimit); velocidade máxima média que um cluster pode ter (avgSpeed); tempo mínimo para geração de clusters (minTime). O Quadro 8 mostra o espaço de busca utilizado no processo de calibração.

Quadro 8 - Espaço de busca utilizado para o CB-SMoT no processo de calibração.

Limites	**Espaço de Busca para cada Parâmetro do CB-SMoT**		
	speedLimit	**avgSpeed**	**minTime**
Limite inferior	0,7	0,3	7200 (2hrs)
Limite superior	2	1,5	36000 (10hrs)

O algoritmo de clusterização DB-SMoT possui três parâmetros de entrada: variação de direção mínima (minDir); tempo mínimo para geração de clusters

(minTime); tolerância máxima entre os pontos da trajetória (maxTol). A Quadro 9 mostra o espaço de busca utilizado no processo de calibração.

Quadro 9 - Espaço de busca utilizado para o DB-SMoT no processo de calibração.

Limites	**Espaço de Busca para cada Parâmetro do DB-SMoT**		
	minDir	**minTime**	**maxTol**
Limite inferior	0,5	21600 (6 horas)	2
Limite superior	30	86400 (24 horas)	7

Para o este processo de otimização, foram criadas 12 configurações, como pode ser visto na Quadro 10. Essas configurações foram escolhidas para avaliar a influência da quantidade de indivíduos por população e do operador de *crossover* nestes algoritmos de clusterização.

Quadro 10 - Configurações utilizadas no processo de calibração dos algoritmos de clusterização.

Configurações	**Algoritmo de clusterização**	**Método de *crossover***	**Quantidade de indivíduos por população**
conf_1	CB-SMoT	BLX Alpha	100
conf_2	CB-SMoT	BLX Alpha	300
conf_3	CB-SMoT	Linear	100
conf_4	CB-SMoT	Linear	300
conf_5	CB-SMoT	BLX Alpha	600
conf_6	CB-SMoT	Linear	600
conf_7	DB-SMoT	BLX Alpha	100
conf_8	DB-SMoT	BLX Alpha	300
conf_9	DB-SMoT	Linear	100
conf_10	DB-SMoT	Linear	300
conf_11	DB-SMoT	BLX Alpha	600
conf_12	DB-SMoT	Linear	600

Os resultados produzidos pela execução de cada uma das configurações podem ser vistos na Tabela 9. A coluna com o valor dos parâmetros possui os valores (speedLimit, avgSpeed, minTime) para o CB-SMoT (configurações de 1 a 6) e (minDir, minTime, maxTol) para o DB-SMoT (configurações de 7 a 12). O resultado de classificação perfeita pode ser visto na segunda linha da Tabela 9, ou seja, a configuração que acha os stops e moves de forma perfeita possuem o valor da função de aptidão igual a 82 (encontrar os 82% de instâncias positivas), que acontece quando o TP_rate é igual a 1 e o FP_rate é igual a 0.

Tabela 9 - Valores encontrados para os componentes da matriz de confusão da análise ROCCH no processo de calibração.

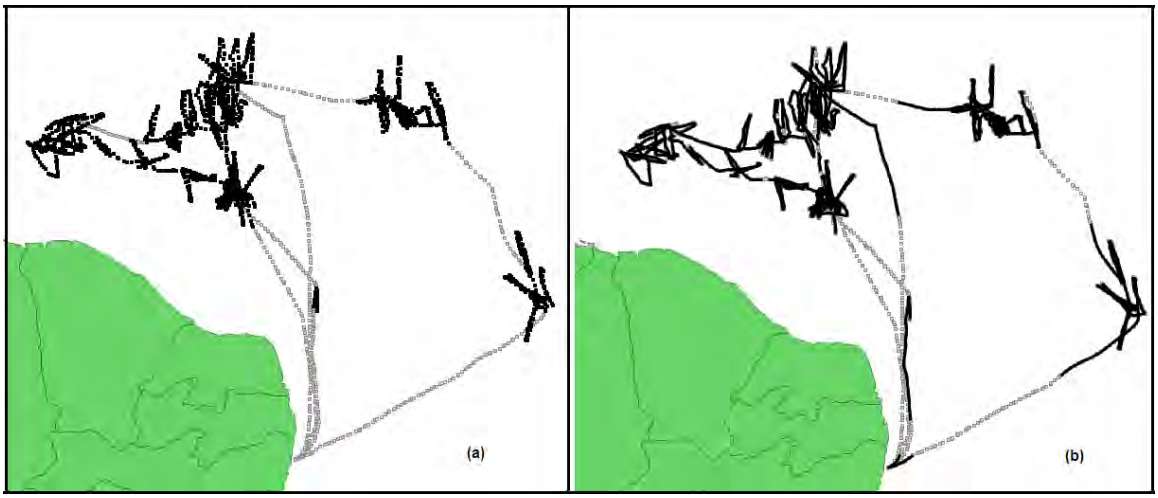
Configurações	Valor dos parâmetros	TP	FP	TN	FN	TP rate	FP rate	Função de aptidão
Parâmetros ideais	(X,X,X)	2229	0	488	0	1.000	0.000	82.000
conf_1	(1.531, 1.162, 21786.744)	2109	64	424	120	0.946	0.131	75.224
conf_2	(1.766, 0.970, 11849.429)	2118	55	433	111	0.950	0.112	75.887
conf_3	(1.875, 0.975, 10800.0)	2114	55	433	115	0.948	0.112	75.740
conf_4	(1.531, 1.162, 21786.744)	2109	64	424	120	0.946	0.131	75.224
conf_5	(1.533, 1.186, 34999.626)	2120	61	421	109	0.951	0.137	75.518
conf_6	(1.875, 0.975, 10800.0)	2114	55	433	115	0.948	0.112	75.740
conf_7	(0.5, 86400.0, 2.0)	2223	178	310	6	0.997	0.364	75.213
conf_8	(2.382, 44105.083, 4.0)	2056	10	478	173	0.922	0.020	75.266
conf_9	(2.075, 47149.931, 5.0)	2183	121	367	46	0.979	0.247	75.844
conf_10	(2.434, 78261.022, 6.0)	2109	74	414	120	0.946	0.151	74.855
conf_11	(1.929, 71324.348, 5.0)	2189	143	345	40	0.982	0.293	75.253
conf_12	(0.5, 83314.284, 3.285)	2229	194	294	0	1.000	0.397	74.844

Como pode ser visto na Tabela 9, todas as configurações utilizadas no processo de calibração encontraram valores para a função de aptidão em torno do valor 75. Esse resultado mostra que as configurações adotadas neste experimento obtiveram bons resultados no processo de otimização de parâmetros, já que o valor perfeito é exatamente 82 (82% de instâncias positivas). Dessa forma, para a atividade de pesca, não se pode inferir qual dos algoritmos de clusterização é melhor para fazer a classificação da trajetória em *stops* e *moves*, pois o melhor valor de função de aptidão encontrado pelo CB-SMoT foi de 75.887 e o do DB-SMoT foi de 75.844. Para afirmar com certeza qual

dos dois algoritmos de clusterização é melhor para este domínio, seriam necessários mais dados de pesca pelágica. Até o momento, não foram obtidos mais dados para este domínio. Futuramente, caso se consiga mais dados deste tipo, pode-se executar este experimento novamente e novas conclusões podem ser tiradas a respeito do melhor algoritmo de clusterização para o domínio de pesca pelágica.

Para ilustrar o melhor resultado encontrado, obtido pela configuração 2 e que utiliza o algoritmo de clusterização CB-SMoT, são mostradas a classificação perfeita na Figura 38(a) e o melhor resultado encontrado na Figura 38(b) pelo processo de otimização.

Figura 38 - Melhor resultado obtido pelo processo de otimização para os algoritmos de clusterização de objetos móveis. (a) classificação perfeita. (b) melhor resultado obtido pelo processo de otimização com o algoritmo CB-SMoT.



Neste experimento, foi mostrada a otimização dos parâmetros de entrada de dois algoritmos (i.e CB-SMoT e DB-SMoT) de clusterização de pontos de trajetória de objetos móveis. A partir do uso da análise ROC na função de aptidão do AG e da modelagem cromossômica do problema, foi realizado um processo de otimização com 12 configurações geradas para a ferramenta X-GAT. Os resultados não foram conclusivos a respeito de qual algoritmo de clusterização é melhor para o domínio de pesca pelágica, pois os melhores resultados dos dois métodos ficaram muito próximos, sendo necessária assim a aquisição de mais dados deste domínio para que esta conclusão seja obtida. Entretanto, esta abordagem é o primeiro passo para inclusão deste tipo de abordagem na área de trajetórias de objetos móveis.

Caso não existissem alternativas como X-GAT, seria necessária a implementação completa de um AG, com todas as dificuldades de validação e testes desta heurística de busca. Desta forma, utilizar uma ferramenta como o X-GAT facilita o processo de otimização aqui realizado, pois a partir das configurações geradas não foi necessário qualquer tipo de implementação de código, com exceção da classe de avaliação criada para avaliar os indivíduos deste experimento.

Capítulo

5

Considerações Finais

Neste capítulo, apresentamos as contribuições realizadas com o trabalho e discutimos os resultados obtidos até agora pela ferramenta X-GAT. Comentamos também sobre dificuldades encontradas, trabalhos futuros e, por fim, mostraremos as conclusões da dissertação.

5.1 Discussão

Uma das dificuldades encontradas ao se construir a ferramenta X-GAT foi, sem dúvidas, a grande quantidade de variações e modificações que podem ser aplicadas em um AG. Além disso, o núcleo do código do AG clássico, se comparado com diversos outros AGs, é bastante modificado, sendo assim difícil manter um mesmo fluxo de execução para a implementação dessas diversas modificações. Com o objetivo de não só prover AGs prontos para ser executados, esta ferramenta foi modelada para ser também um *framework*, podendo assim auxiliar na criação de outras variações do AG clássico. Essa questão aumenta a complexidade para a modelagem da ferramenta de otimização, pois seria difícil criar uma estrutura que fosse capaz de executar qualquer AG. Assim, foi mantido o fluxo de execução do AG clássico, sendo apenas mapeadas as variações de cada bloco (i.e métodos de seleção, *crossover*, mutação...) do algoritmo e implementado um AG que seguia o mesmo fluxo de execução do AG clássico, que foi o AG (μ+λ).

O X-GAT, quando comparado com outras soluções disponíveis possui três diferenciais:

1. Possui AGs implementados e prontos para uso, necessitando apenas que se crie uma configuração que informe as funcionalidades a serem utilizadas no processo de otimização;

2. Fornece um *framework* para auxiliar a implementação de outros AGs, podendo ser re-aproveitadas diversas partes do algoritmo, bem como a modelagem do genótipo dos parâmetros a serem otimizados;
3. A utilização de XML nos arquivos de entrada e saída possibilita a construção de sistemas heterogêneos.

Ao se comparar o X-GAT com a ferramenta Java GALib, conforme mostrado na Quadro 2, fica evidente sua superioridade quanto as questões de extensibilidade, pois o Java GALib possui apenas uma implementação de um AG. Ao se comparar com a ferramenta JAGA, o X-GAT possibilita a criação de sistemas heterogêneos, possibilidade não oferecida pelo JAGA. Entretanto, diferente do X-GAT, a ferramenta JAGA possui uma série de operadores genéticos (i. e., operadores específicos para seqüências de proteínas) que podem ser aplicados com robustez em problemas biológicos, funcionalidades não providas pelo X-GAT e que o framework desenvolvido não oferece facilidades para que estas funcionalidades sejam implementadas. Além disso, o JAGA fornece algumas ferramentas de análise que podem ser usadas para verificar estatísticas de população (i. e., pior nota de aptidão, nota de aptidão média, desvio padrão...). Já o JGAP não possui algoritmos prontos para uso, sendo necessário assim que o usuário entenda como funciona um AG e utilize as funcionalidades da ferramenta para executar uma otimização. Entretanto, o JGAP fornece em sua API uma forma de criar *grid computacionais* para aumentar a rapidez de execução de um AG. Outra possibilidade interessante, é que esta funcionalidade possibilita a criação de AGs paralelos, que são outras variações de AGs, sendo estes executados em mais de um computador e com uma série de outros conceitos que envolvem sua concepção.

Com relação aos testes, foram otimizadas algumas funções matemáticas para as quais o valor ótimo era conhecido. Os resultados obtidos tanto pelo AG clássico quanto pelo AG (μ+λ) mostraram que ambos tiveram configurações que conseguiram atingir os valores ótimos dos problemas de otimização. Um resultado evidente é o de que o *crossover* linear combinado com a inicialização por *grid* em praticamente todas as simulações conseguiram obter esse valor ótimo. Entretanto, o *crossover* linear é determinístico por natureza, isso significa que filhos gerados pelos mesmos pais serão sempre idênticos, fazendo com que a variabilidade genética seja baixa durante um processo de otimização. Não houve grandes diferenças no uso dos dois tipos de AGs com relação à obtenção do valor ótimo, mas como pode ser observado na Seção 4.1, o AG (μ+λ) possui um tempo de execução ligeiramente maior se comparado ao AG clássico.

Um resultado que pode parecer estranho à primeira vista deve ser levado em consideração (i.e. configurações 7 e 8 na otimização da função de Goldstein-price): algumas configurações com valor menor de indivíduos por população encontraram o ótimo global, enquanto algumas configurações que possuíam uma quantidade maior de indivíduos por população e utilizavam praticamente as mesmas funcionalidades não obtiveram este valor ótimo. Como dito na Seção 2.2, os AGs não garantem que o ótimo global será sempre encontrado, pois os AGs, na verdade são uma heurística de busca para ótimos globais. Os indivíduos que foram cruzados durante o processo de evolução convergiram para o valor ótimo com uma quantidade de indivíduos menor. Por outro lado, nas configurações com uma quantidade de indivíduos por população maior, não aconteceram cruzamentos que levaram a este valor ótimo. Este resultado não torna o X-GAT uma implementação inválida de AGs, pois resultados próximos do ótimo sempre foram encontrados. O que fica claro é que qualquer processo de otimização não depende apenas da configuração adotada, e que a execução da mesma configuração poderá produzir resultados diferentes após o término de uma otimização.

No segundo experimento, realizamos a calibração automática de parâmetros de entrada de modelos hidrológicos, usados com freqüência na área de engenharia para tomada de decisão no que diz respeito à gestão de águas. Foram realizados dois experimentos, um com um modelo simples (*tank model*) e outra com um modelo mais complexo (WESP). Em ambos a ferramenta X-GAT conseguiu obter resultados satisfatórios, pois diante dos critérios estabelecidos para avaliar cada um deles, seus valores foram considerados satisfatórios. Dessa forma, fica mostrada a viabilidade da utilização da ferramenta X-GAT em outros modelos hidrológicos.

No terceiro experimento foram mostrados os resultados do processo de otimização de parâmetros de entrada de algoritmos de clusterização de pontos de trajetórias de objetos móveis aplicados no domínio de pesca marítima, com o objetivo de verificar qual dos algoritmos (CB-SMoT ou DB-SMoT) era melhor para este domínio. Para o cálculo da função de aptidão foi utilizada a análise ROCCH, sendo então criadas e executadas 12 configurações com a ferramenta X-GAT para realizar este processo de otimização. Neste experimento, foi encontrado um resultado interessante, pois os dois algoritmos de clusterização utilizados obtiveram resultados para a função de aptidão muito próximos, não sendo assim possível dizer com precisão qual dos dois algoritmos é melhor para gerar *stops* e *moves* no domínio de pesca marítima. Este experimento tinha o objetivo de mostrar que a ferramenta X-GAT era capaz de ser aplicável em mais um

domínio, o que foi realizado e mostrado ao longo da análise do experimento com dados de pesca marítima.

5.2 Trabalhos Futuros

O trabalho pode ser expandido em diversas direções. Citaremos aqui as que achamos mais relevantes:

1. Implementar mais operadores genéticos para áreas específicas, criando-se várias extensões da ferramenta para domínios de aplicação específicos;
2. Expandir a ferramenta para outras heurísticas de busca, utilizando o mesmo processo de modelagem, construção e implementação;
3. Possibilitar a execução de AGs paralelos, diminuindo assim o tempo de processamento de uma otimização e aumentando a possibilidade de novas implementações de AGs;
4. Construção de um ambiente de testes para otimização com AGs, possibilitando assim a análise dos melhores operadores genéticos para um determinado domínio de aplicação.

5.3 Conclusões

Consideramos que avançamos na criação de uma ferramenta de otimização com AGs de propósito geral, pois como pode ser visto na Quadro 11, a ferramenta X-GAT é a única a ser:

- **Genérica:** qualquer problema de otimização pode ser resolvido pela ferramenta X-GAT desde que seja feita uma configuração adequada;
- **Configurável:** a partir de um arquivo XML de entrada, várias funcionalidades distintas podem ser utilizadas no processo de otimização;
- **Extensível**: a ferramenta foi modelada para suportar a adição de novas funcionalidades de uma forma simples e prática a partir do uso de padrões de projeto de software;
- **Portável:** como a ferramenta foi implementada com a linguagem de programação Java, qualquer sistema operacional que possua a máquina virtual Java instalada corretamente é capaz de executar otimizações com o X-GAT;

- **Permite a construção de sistemas heterogêneos**: a entrada e saída de dados da ferramenta X-GAT é realizada a partir de arquivos XML. Desta forma, aplicações em linguagens de programação capazes de processar dados em XML podem usar a ferramenta X-GAT;
- **Algoritmos prontos para executar:** diferente de vários frameworks disponíveis para download, a ferramenta X-GAT já possui AGs prontos para uso, bastando apenas serem escolhidas e configuradas as funcionalidades a serem utilizadas no processo de otimização;
- **Documentação**: a ferramenta X-GAT vem com vários exemplos de uso prontos para serem executados por qualquer usuário que faça o seu download. Além disso, todos os AGs implementados, bem como as funcionalidades disponibilizadas para serem usadas no framework possuem documentação detalhada.

Quadro 11 - Ferramentas de otimização com AGs de propósito geral após a construção da ferramenta X-GAT.

	Genérica	Configurável	Extensível	Portabilidade de Sistema Operacional	Construção de sistemas heterogêneos	Algoritmos prontos para executar	Documentação	Execução em *grids* computacionais
JGAP	O	O	O	O	O	X	O	O
Java GALib	O	O	X	O	X	O	X	X
JAGA	O	O	O	O	X	O	X	X
X-GAT	O	O	O	O	O	O	O	X

Legenda: O – A ferramenta contém a característica.

X – A ferramenta não contém a característica.

Entretanto, a ferramenta X-GAT não foi preparada para permitir otimizações em *grids* computacionais (funcionalidade suportada apenas pela ferramenta JGAP), ficando a implementação desta funcionalidade como um trabalho a ser realizado nas próximas versões desta ferramenta.

O X-GAT é uma ferramenta livre e de código aberto, possibilitando assim que o seu código possa ser alterado, melhorado e expandido pela comunidade acadêmica. A ferramenta está disponível para download no link: http://x-gat.sourceforge.net. O X-

GAT foi disponibilizado sob a licença GNU Library or Lesser General Public License (LGPL). Isso significa que a ferramenta X-GAT pode ser utilizada pela academia e pela indústria sem restrições, desde que o trabalho fique disponível para toda a comunidade como uma biblioteca. A partir do uso dessa licença, pretendemos difundir a sua utilização em vários meios e atrair novos colaboradores para ampliar a quantidade de funcionalidades a serem disponibilizadas para o público.

A aplicação da ferramenta X-GAT em alguns problemas de otimização com objetivo de validar os AGs implementados, bem como testar as funcionalidades implementadas, concluiu o ciclo de produção da primeira versão da ferramenta. Podemos afirmar que os objetivos levantados na Seção 1.2 foram atendidos, pois estes foram levados em consideração durante a concepção, construção e validação da ferramenta, resultando em uma ferramenta de otimização com AGs livre e diferente das existentes no mercado.

A partir dos experimentos realizados foi mostrado que a partir de uma configuração e utilização adequada, qualquer problema de otimização contínua no qual os AGs possam ser aplicados pode ser avaliado pela ferramenta X-GAT, sendo este trabalho o início de uma ferramenta para otimização de qualquer tipo de problema nas mais diferentes áreas da ciência.

REFERÊNCIAS

Alexander, C.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl-King I.; Angel S. (1977). A Pattern Language. Oxford University Press, NewYork.

Arnold, K., *et al.* (2005). Java(Tm) Programming Language: Addison-Wesley Professional. (Java Series)

Bramlette, M. F., Cusic, R. (1989), A Comparative Evaluation of Search methods Applied to Parametric Design of Aircraft. In Proceedings of the Thrid International Conference on Genetic Algorithms, San Mateo, California. Morgan Kaufmann.

Bramlette, M. F. (1991). Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization. In Belew, R. K., and Booker, L. B., eds., Proceedings of the Fourth International Conference on Genetic Algorithms, 100-107. Morgan Kaufmann

Bernal, A.; Ramirez, M.A.; Castro, H.; Walteros, J.L.; Medaglia, A.L. (2009) JG2A: A Grid-Enabled Object-Oriented Framework for Developing Genetic Algorithm. Systems and Information Engineering Design Symposium. pp 67-72. ISBN: 978-1-4244-4531-8. doi: <10.1109/SIEDS.2009.5166157>

Booch et. al (2000) The Unified Modeling Language User Guide, First Edition.. Addison-Wesley.

Bogorny, V. & Wachowicz, M. (2008). A Framework for Context-Aware Trajectory Data Mining. Springer.

Cadier, E., Freitas, B.J., & Leprun, J.C. (1983) Bacia Experimental de Sumé. Instalação e primeiros resultados. SUDENE, Recife, Pernambuco, Brasil, p. 87.

Celeste, A. B., Suzuki, K., & Kadota, A. (2004). Genetic algorithms for real-time operation of multipurpose water resource systems. Journal of Hydroinformatics, vol. 6; Part 1, pp. 19–38.

Christiani, S. (2009). Creation: A Framework for Artificial Evolution Using Complexity Matching. Dissertação de mestrado, Departamento de Informática da Universidade de Zurique – Suíça.

Croley, T. E., II, (1982) Unsteady overland sedimentation. J. of Hydrology, Elsevier 56, 325–346.

Darwin, C. R., (1859) On The Origin of Species by Means of Natural Selection. London, U.K.: J. Murray, ISBN 0-517-12320-7.

Davis, S. S. (1978) Deposition of nonuniform sediment by overland flow on concave slopes. MSc Thesis, Purdue University, West Lafayette, IN, USA.

De Jong K. A. (1975) Analysis of the Behavior of a Class of Genetic Adaptive Systems. Dissertação de Doutorado, Department of Computer and Communication Science, University of Michigan, Estados Unidos.

Diniz, L. S. (1999). Calibragem de modelos hidrológicos. In: Sistemas Inteligentes: aplicações a recursos hídricos e ciências ambientais. Org. por Galvão, C.O. e Valença, M.J.S., Ed. ABRH. UFRGS, Porto Alegre – RS, pp. 151-164.

Dixon, L. C. W. & Szego, G. P. (1978). The optimization problem: An introduction. In: Towards Global Optimization II (North Holland).

Einstein, H. A. (1968) "Deposition of suspended particles in a gravel bed". J. of the Hydraulic Division, Proc. ASCE, 94(HY5), 1197–1205.

Eshelman LJ & Schaffer DJ (1993) Real-coded genetic algorithms and interval-schemata, Foundations of genetic algorithms 2. Morgan Kaufmann, pp 187–202

Fawcett, T. (2006) An introduction to ROC graphs. Pattern Recognition Letters, vol. 27, no. 8, pp 861-874, 2006.

Foster, G. R. (1982) Modeling the erosion process. In: Hydrologic modeling of small watersheds (ed. by C. T. Haan, H. P. Johnson & D. L. Brakensiek), Am. Soc. Agr. Eng., 295–380.

Foster, I., Tuecke, S. (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations.

Furuya, H. & Haftka, R. T. (1993), Genetic Algorithms for Placing Actuators on Space Structures. In Proceedings of the 5th International Conference on Genetic Algorithms. Morgan Kaufmann. pp - 536-542

Gamma, E., Helm, R., Johnson R., Vlissides, J. (1999), Design Patterns: Elements of Reusable Object-Oriented Software. 1ª Edição, 416 p.

Goldstein, A. A. & Price, I. F. (1971) On descent from local mínima. Math. Comput., vol 25, nro 115.

Graves, M. (2003). Projeto de Banco de Dados com XML. Ed. Pearson, São Paulo – SP, 518 p.

Green, W. H. & Ampt, G. A. (1911) Studies on soil physics, I. The flow of air and water through soils. J. Agr. Sci., 4(1), 1–24.

Holland, J. H., (1975). Adaptation in Natural and Artificial Systems, MIT Press.

Hrstka O. & Kucerová A. (2004)**.** Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. Advances in Engineering Software, vol 35, pp. 237-246.

JAGA (2004). Java API for Genetic Algorithms – Disponível em: <http://www.jaga.org/index.html> (último acesso em agosto de 2010)

Janikow, C. Z. & Michalewicz, Z. (1991) An experimental comparison of binary and floating point representations in genetic algorithms. In: Proc. of the Fourth International Conference on Genetic Algorithms (San Diego, CA, USA), Morgan Kaufmann, 31–36.

JGAP (2002). Java Genetic Algorithms and Genetic Programming Package – Disponível em:< http://jgap.sourceforge.net/> (último acesso em agosto de 2010).

Johnson, R. E. (1997) Frameworks = Components + Patterns. How frameworks compare to other objetct oriented reuse techniques. Communicationof ACM, vol.40, no. 10, October 1997, pp. 39-42.

Larman, C. (2007) Utilizando UML e Padrões, Bookman, 3ª Edição , 695 p.

Lopes, V. L. (1987). A numerical model of watershed erosion and sediment yield. Ph.D. Thesis. University of Arizona. Tucson, Arizona (USA).

Lopes, V.L., & Lane, L.J. (1988). Modeling sedimentation processes in small watersheds. In Sediment Budgets, ed. by M. P. Bordas & D.E. Walling, IAHS Publication no. 174, 497–508.

Madeiro, S. S. (2010) Buscas Multimodais por Cardumes baseados em Densidade. Dissertação de mestrado. Universidade de Pernambuco – Escola Politécnica de Pernambuco, Departamento de Sistemas e Computação.

Manso, J. A. ; Times, V. C. ; Oliveira, G. ; Alvares, Luis Otavio ; Bogorny, V. (2010). DB-SMoT: A Direction- Based Spatio-Temporal Clustering Method. In: IEEE International Conference on Intelligent Systems (IS), Londres.

Matou, K., Lep, M., Zeman, J., Sejnoha M. (2000). Applying genetic algorithms to selected topics commonly encountered in engineering practice. Computional Methods Applied Mechanical Engineering 190, pp 1629-1650.

Martinez, E. Z., Louzada-Neto, F. & Pereira, B. B. (2003) A Curva ROC para Testes Diagnósticos. Cadernos Saúde Coletiva, Rio de Janeiro; 11(1):7-31.

Mehta, A. J. (1983) Characterization tests for cohesive sediments. In: H. T. Shen (editor), Proc. of the Conference on Frontiers in Hydraul. Engng. ASCE/MIT, Cambridge, Mass., 79–84.

Kisi O. & Guven A. (2010). A machine code-based genetic programming for suspended sediment concentration estimation. Advances in Engineering Software vol. 41 pp. 939-945. doi:<10.1016/j.advengsoft.2010.06.001>.

Palma, A. T. , Bogorny V., Kuijpers B., Alvares. L. O. (2008). A Clustering-based approach for Discovering Interesting Places in Trajectories. Proceedings of the 2008 ACM Symposium on Applied computing. Fortaleza, Ceará, Brasil.

Paperin, G. (2008). Using Holey Fitness Landscapes to Counteract Premature Convergence in Evolutionary Algorithms. Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation, Atlanta, USA.

Paperin, G., Michalas A. (2004). Evolving a Multiplexer using Genetic Programming. Technical Report. Disponível em:<http://www.jaga.org/download.html#extras> (último acesso em agosto de 2010).

Prati, R. C., Batista, G. E. A. P. A.; Monard, M. C. (2008) Curvas ROC para a avaliação de classificadores. Revista IEEE América Latina, São Paulo.

Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function, The Computer Journal 3: 175–184, doi:<10.1093/comjnl/3.3.175>, MR0136042, ISSN 0010-4620.

Salas, J.D. (1993) Analysis and modeling of hydrologic time series, in Handbook of Hydrology, edited by D. R. Maidment, Chap. 19, pp. 19.1-19.72, McGraw-Hill Inc., New York, USA.

Santos, C. A. G., Suzuki, K., Watanabe, M. & Srinivasan, V.S. (1994) Optimization of coefficients in runoff-erosion modeling by Standardized Powell method, J. of Hydrosci. Hydraul. Engng, 12(1), 67–78. .

Santos, C. A. G., Srinivasan, V. S., Suzuki, K., & Watanabe, M. (2003). Application of an optimization technique to a physically based erosion model. Hydrological Processes, vol. 47, pp. 989–1003, doi: <10.1002/hyp.1176>.

Santos, J. C., Oliveira, J. R. F., Dutra, L. V., Sant'Anna, S. J. S., Rennó, C. D. (2007). Seleção de atributos usando algoritmos genéticos para classificação de regiões. Anais XIII Simpósio Brasileiro de Sensoriamento Remoto, Florianópolis, Brasil, INPE, p. 6143-6150.

Senouci, A. B., Al-Ansari, M. S. (2009). Cost optimization of composite beams using genetic algorithms. Advances in Engineering Software 40 (2009) pp.1112–1118. doi:<10.1016/j.advengsoft.2009.06.001>.

Sivanandam, S. N., Deepa, S. N. (2008). Introduction to Genetic Algorithms. Springer. 453 pags

Sommerville, I. (2007) Engenharia de Software. Editora Addison Wesley. 8 edição.

Spaccapietra, S., Parent C., Damiani M. L., Macedo J. A., Porto F., Vangenot C. (2007). A Conceptual View on Trajectories. Data and Knowledge Engineering (DKE) (65): 126-146.

Sugawara, M. (1979). Automatic calibration of the tank model. Hydrological Science Bulletin, Vol. 24 (3).

Thompson A., Layzell P., & Zebulum R. S. (1999). Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution. IEEE Transactions on Evolutionary Computation, vol. 3, no. 3, September.

Törn, A. & Zilinskas, A. (1989) Global Optimization. Lecture Notes in Computer Science, Nº 350, Springer-Verlag, Berlin.

Anexo

A

Sintaxe para Codificação de Funções com o JEP

Operação	**Símbolo**
Elevar	^
Mais e menos unário	+x,-x
Resto da divisão	%
Divisão	/
Multiplicação	*
Soma	+
Subtração	-
Menor igual, maior igual	<=, >=
Menor, maior	<, >
Diferente, igual	!=, ==

Funções	**Símbolo**
Seno	sin(x)
Cosseno	cos(x)
Tangente	tan(x)

Arco seno	asin(x)
Arco cosseno	acos(x)
Arco tangente	atan(x)
Arco tangente (com 2 parâmetros)	atan2(y, x)
Seno hiperbólico	sinh(x)
Cosseno hiperbólico	cosh(x)
Tangente hiperbólica	tanh(x)
Seno hiperbólico inverso	asinh(x)
Cosseno hiperbólico inverso	acosh(x)
Tangente hiperbólica inversa	atanh(x)
Logaritmo natural	ln(x)
Logaritmo na base 10	log(x)
Exponencial (e^x)	exp(x)
Valor absoluto	abs(x)
Número randômico (entre 0 e 1)	rand()
Módulo	mod(x,y) = x % y
Raiz quadrada	sqrt(x)