Ribamar Loura do Carmo

Uma meta-heurística baseada em Algoritmo Genético para Seleção de Oficiais de Justiça por Distritos

São Luís, Brasil

2016

Ribamar Loura do Carmo

Uma meta-heurística baseada em Algoritmo Genético para Seleção de Oficiais de Justiça por Distritos

Dissertação apresentada ao curso de Engenharia da Computação - ENGCOMP, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia da Computação na Universidade Estadual do Maranhão.

Universidade Estadual do Maranhão - UEMA

Pró-Reitoria de Pesquisa e Pós-Graduação

Departamento de Engenharia da Computação

Orientador: Prof. Dr. Omar Andres Carmona Cortes

Coorientador: Prof. Dr. Fernando Jorge Cutrim Demétrio

São Luís, Brasil

2016

Ribamar Loura do Carmo

Uma meta-heurística baseada em Algoritmo Genético para Seleção de Oficiais de Justiça por Distritos

Dissertação apresentada ao curso de Engenharia da Computação - ENGCOMP, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia da Computação na Universidade Estadual do Maranhão.

Trabalho aprovado. São Luís, Brasil, ___ de _____ de 2016:

Prof. Dr. Omar Andres Carmona Cortes
Orientador

Prof. Dr. Fernando Jorge Cutrim Demétrio
Coorientador

Prof. Msc. Josenildo Costa da Silva
Convidado

Prof. Dr. Bruno Feres
Convidado

São Luís, Brasil
2016

Este trabalho é dedicado a minha família,
cujos valores éticos e princípios cristãos são legados de meus pais.

Agradecimentos

Agradeço, primeiramente, a Deus por ter me guiado por este caminho de superação diária.

Aos professores Omar Carmona e Fernando Demétrio e a todos aqueles que contribuíram para que a produção desse trabalho acadêmico fosse possível.

A meus pais, Cipriano e Francisca (in memória), que através de seus preceitos cristãos, suas índoles e valores morais ilibados, criaram e mantiveram unida a grande família Loura.

A minha esposa, Arlety, e minhas filhas, Heloísa e Letícia, pelo apoio incondicional e motivação constante, meu muito obrigado por me suportarem nos momentos mais difíceis.

Aos meus sobrinhos: Aldivam, Odinea, Marcus, Everaldo e Francisco Antônio e, também aos demais familiares e amigos por acreditarem na minha superação.

Aos membros da banca pela disposição e análise deste documento e pelas contribuições propostas.

À FAPEMA - Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão por patrocinar essa pesquisa[1].

[1] <www.fapema.br/>

"...e conhecereis a verdade, e a verdade vos libertará."
(Bíblia Sagrada, João 8:32)

Resumo

Este trabalho apresenta uma proposta para resolver o Problema de Seleção de Oficiais de Justiça por Distritos (PSOJ/D) nas centrais de mandados no Estado do Maranhão-Brasil. A solução, chamada de AGSOJ/D, é baseada na meta-heurística conhecida como Algoritmo Genético (AG), sendo a solução completa para o PSOJ/D um *software* baseado na *Web*, chamado AGSOJ/D. Foram realizados vários experimentos considerando dois cenários. O primeiro representando a central de mandados na cidade de Imperatriz com 6 distritos e 27 oficiais de justiça. O Segundo retratando a central de mandados na cidade de São Luís com 16 distritos e 84 oficiais de justiça. Os resultados do AGSOJ/D são comparados com três abordagens: seleção manual atualmente em uso, AGSOJ/D-X com operador de cruzamento e *Simulated Annealing (SA)*. Os resultados mostraram que o AGSOJ/D seleção-mutação criou soluções melhores que o processo manual, e é mais estável que o AGSOJ/D-X e o SA, especialmente no segundo cenário, em que o espaço de busca é 2^{1344} contendo $\approx 1,10 * 10^{89}$ possibilidades de soluções válidas. Além disso, o Apêndice A deste trabalho contempla o projeto de *software* do AGSOJ/D com os requisitos funcionais, não-funcionais, casos de usos e diagramas de classes, além de tecnologias e ferramentas utilizadas no desenvolvimento do *software*.

Palavras-chave: oficial de justiça, np-completo, meta-heuristica, algoritmo genético, têmpera simulada.

Abstract

This work proposes a solution to the problem of selecting bailiffs by districts (PSB/D) at the central of warrants in Maranhão, Brazil. The solution, named AGSOJ/D, is based on a meta-heuristic called Genetic Algorithms (GA). The complete solution to the PSB/D problem is a Web-based software called GAPSB/D. Were conducted various experiments to prove its applicability considering two scenarios. The first one represents the central of warrants in the city of Imperatriz devised by six districts and 27 bailiffs. The second one depicts the central of warrants in the city of São Luís composed by 16 districts and 84 bailiffs. The results of the GAPSB/D are compared against three approaches: the manual selection currently in use, GAPSB/D-X with crossover operator, and Simulated Annealing (SA). Results have shown that GAPSB/D could create solutions as good as the manual process, and it is more stable than GAPSB/D-X and SA, especially in the second scenario, in which the size of the search space is 2^{1344} containing $\approx 1,10 * 10^{89}$ possibilities of feasible solutions. Moreover, the Appendix A of this work includes the software design of the GAPSB/D, which contains both functional and non-functional requirements, use cases, and class diagrams. Further, presents the technologies and tools used in the development of the software GAPSB/D.

Keywords: bailiff, np-hard, meta-heuristic, genetic algorithm, simulated annealing.

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Lista de abreviaturas e siglas

AGH	Algoritmo Genético Híbrido
AGSOJ/D	Algoritmo Genético para Seleção de Oficiais de Justiça por Distrito
CASE	do inglês, Computer-Aided Software Engineering (Engenharia de Software auxiliada por computador)
CGJ	Corregedoria Geral de Justiça
IDE	do inglês, Integrated Development Environment (Ambiente de Desenvolvimento Integrado)
GA	do inglês, Genetic Algorithm (Algoritmo Genético - AG)
SA	do inglês, Simulated Annealing (Têmpera Simulada - TS)
Java EE	do inglês, Java Enterprise Edition (Java Edição Empresarial)
MVC	do inglês, Model-View-Controller (Modelo-Visão-Controlador)
OJ	Oficial de Justiça
PAE	Problema de Atribuição de Escalas
PAS	Problema de Alocação de Salas
PMM	Problema Multidimensional da Mochila
PPT	Problema de Programação de Tripulantes
PSOJ/D	Problema da Seleção de Oficiais de Justiça por Distrito
PRV	Problema de Roteamento de Veículos
UC	Use Case (Caso de uso em português)

Sumário

1 Introdução

Problemas do mundo real normalmente são complexos e na maioria das vezes algoritmos enumerativos são impossíveis de serem aplicados devido às restrições e ao tamanho do espaço de busca. Dentre esses problemas, pode-se destacar o Problema de Seleção de Oficiais de Justiça por Distritos (PSOJ/D), que é disciplinado pelo Provimento 18/2011 da Corregedoria Geral de Justiça do Estado do Maranhão (CGJ-MA) (PROVIMENTO 18, 2011). O Art. 2º desse provimento diz que a alocação dos oficiais de justiça nas respectivas áreas (distritos) dar-se-á por sorteio (distribuição) trimestral, sendo realizado eletronicamente pela TI (Tecnologia da Informação) do Tribunal de Justiça.

Atualmente, a seleção é feita através de uma planilha eletrônica Microsoft Excel, sendo o processo de distribuição feito de forma manual de modo a atender às restrições do problema; processo este que leva em torno de 2 dias e pelo menos 2 funcionários, concentrados nessa atividade, para realizar as 4 distribuições referentes a 1 ano, sendo distribuição a atividade de selecionar os oficiais de justiça por distritos. Além disso, o espaço de busca é muito grande, o cenário 1 que representa o problema da central de mandados de Imperatriz é 2^{162} com $\approx 6,70 * 10^{15}$ possibilidades de soluções válidas e o cenário 2 que representa o problema da central de mandados de São Luís2^{1344} com $\approx 1,10 * 10^{89}$ possibilidades de soluções válidas. O PSOJ/D é um problema NP-Completo (ou seja, pode ser reduzido em tempo polinomial) como é demonstrado no Capítulo 3, o que torna atraente sua resolução através da utilização de uma meta-heurística.

As meta-heurísticas são técnicas avançadas de busca responsáveis pela obtenção de melhores resultados, de acordo com a literatura e são capazes de explorar uma parte maior do espaço de solução e sair de ótimos locais (REINA, 2012). Logo, *Simulated Annealing* e Algoritmo Genético são exemplos de meta-heurísticas bastante conhecidas da literatura. De fato, os trabalhos correlatos, objetos de pesquisa bibliográfica, envolvem também problemas difíceis de serem tratados, tais como:

- Problema de Programação de Tripulantes - PPT que tradicionalmente tem sua resolução dividida em dois subproblemas: Problema de Determinação das Viagens - PDV e Problema de Atribuição de Escalas - PAE, que embora reduza a sua complexidade não conduz a uma estimativa real de custo e influi na qualidade da solução final. Para este problema, os autores apresentam uma metodologia para modelagem integrada do PPT através de um Algoritmo Genético Híbrido - AGH, associado a um procedimento de busca em profundidade e a um modelo de programação linear inteira, que representa as jornadas viáveis enumeradas, levando em conta as particularidades da legislação brasileira (GOMES; GUALDA, 2011).

- Problema Multidimensional da Mochila - PMM que é um problema clássico de otimização para o qual há vários métodos matemáticos discretos e estocásticos para a sua resolução. Os autores propõem um novo algoritmo inspirado em Evolução Diferencial, a Evolução Diferencial Binária - EDB e compara seus resultados com os resultados alcançados utilizando Algoritmos Genéticos (KRAUSE; PARPINELLI; LOPES, 2012).

- Problema de Roteamento de Veículos - PRV, cuja forma muito comum de se representar é através de grafos ponderados, onde os vértices representam o depósito e os clientes, e as arestas representam os caminhos ligando os diversos vértices entre si. Os pesos das arestas representam os custos de se percorrer os caminhos (HEINEN; OSÓRIO, 2006).

Além dos trabalhos descritos acima, outros como (BHAWNA; KUMAR; BHATIA, 2016), (CORTES; SILVA, 2010), (GOMES; SARAIVA, 2016), (SINGH; SHARMA; VAIBHAV, 2016), (WERNER; FOGARTY, 2015) e (WIJAYANINGRUM; MAHMUDY, 2016) solucionam diversos problemas, mas discorrem sobre problemas completamente diferentes do PSOJ/D. Desse modo, a proposta é resolver o PSOJ/D usando uma nova meta-heurística baseado em Algoritmo Genético (AG) (CORTES et al., 2013), (LINDEN, 2008), (LINDEN, 2012). Na verdade, é usado um algoritmo específico chamado AGSOJ/D seleção-mutação. Mais detalhes sobre AG seleção-mutação podem ser vistos no trabalho de (BÉRARD, 2005). A aplicação final, chamada de AGSOJ/D, foi desenvolvida para Web utilizando a combinação de ferramentas de código aberto, técnicas, modelos e padrões baseados na Engenharia de Software (SOMMERVILLE, 2007) e fundamentado em trabalhos dos últimos cinco anos baseados na meta-heurística de AG.

O foco desse trabalho baseia-se na meta-heurística de Algoritmo Genético, logo os conceitos sobre Algoritmos Evolucionários são apenas para uma melhor compreensão dessa técnica de busca e como estão classificados os AGs.

É nesse contexto que este trabalho modela o PSOJ/D como um problema de otimização de modo que possa ser solucionado baseado na meta-heurística de Algoritmo Genético (AG). Além do mais, não se tem conhecimento de outra forma de solucionar o PSOJ/D que não seja a manual, através e planilhas do Excel, sendo esta uma aplicação nova na área.

A otimização consiste em maximizar o tempo de retorno dos oficiais de justiça a determinado distrito. O tempo é trimestral como determina o Provimento 18/2011 da CGJ-MA. O melhor distrito para um oficial de justiça é aquele que ele ainda não foi selecionado ou aquele em que o oficial de justiça foi selecionado há mais tempo. Mais detalhes sobre como o problema foi otimizado podem ser vistos na Seção 4.1.1.

A solução desse problema através do uso de um *software* utilizando uma meta-heurística baseada em Algoritmo Genético irá otimizar o tempo dos oficiais de justiça nos distritos e facilitar a rotina de trabalho dessas secretarias, haja vista que após a distribuição automatizada, o armazenamento na base de dados também será automático. Atualmente, a distribuição por ser feita em planilhas do Microsoft Excel deve ser digitada manualmente no sistema de acompanhamento processual chamado Themis (Sistema de uso interno, desenvolvido pela TI do Tribunal de Justiça).

Para validar os resultados foram realizadas várias execuções no algoritmo AGSOJ/D seleção-mutação e as saídas foram comparadas com processos anteriores realizados de forma manual, com saídas de outra meta-heurística, o *Simulated Annealing* e também com saídas do AGSOJ/D-X implementado com cruzamento. Essa comparação serviu para fundamentar as discussões e concluir que o projeto de *software* AGSOJ/D é uma solução viável para o PSOJ/D.

A seguir serão apresentados os objetivos: geral e específicos desse trabalho.

1.1 Objetivos

1.1.1 Objetivos Gerais

O objetivo do presente trabalho é criar um novo algoritmo chamado AGSOJ/D seleção-mutação, baseado na meta-heurística de Algoritmo Genético otimizar a seleção de oficias de justiça por distritos nas centrais de mandados de São Luís e Imperatriz do Poder Judiciário do Estado do Maranhão. A solução proposta baseada em AG utiliza a representação binária (LINDEN, 2012) e não permite selecionar um mesmo oficial de justiça em mais de um distrito por distribuição. Além disso, o algoritmo AGSOJ/D seleção-mutação é comparado com o processo manual e também com mais duas abordagens apresentadas no Capítulo 4.

1.1.2 Objetivos Específicos

- Pesquisar trabalhos correlatos dos últimos 5 anos que utilizam Algoritmos Genéticos como proposta de solução para PSOJ/D;
- Elicitar e analisar os requisitos para entender o PSOJ/D;
- Definir o tamanho do espaço de busca e a quantidade de soluções válidas do PSOJ/D;
- Representar o PSOJ/D na meta-heurística de Algoritmo Genético de forma que os indivíduos possam ser avaliados considerando as restrições do PSOJ/D;

- Estudar o *framework* chamado Mentawai para Java que servirá para agilizar o desenvolvimento do protótipo AGSOJ/D, pois esse é o *framework* padrão utilizado no Tribunal de Justiça;
- Criar o projeto de *software* AGSOJ/D com os artefatos e documentação;
- Implementar o protótipo de *software* AGSOJ/D para Web utilizando a linguagem de programação Java e testar a implementação do protótipo de *software*;
- Realizar os experimentos em dois cenários representando as centrais de mandados de Imperatriz e São Luís utilizando o AGSOJ/D seleção-mutação, AGSOJ/D-X com cruzamento e a *Simulated Annealing* (SA), cujos resultados obtidos foram comparados e serviram para discussões e parte da conclusão;
- Comparar os resultados AGSOJ/D seleção-mutação com resultados de distribuições realizadas manualmente.

1.2 Metodologia

A proposta para solução do PSOJ/D que é inspirada na meta-heurística de algoritmo de genético (LINDEN, 2012), (BASGALUPP, 2010), incorpora restrições específicas do Provimento 18/2011 da Corregedoria Geral da Justiça - CGJ-MA. Existem métodos peculiares para gerar a população inicial e reparar indivíduos inválidos. Após a geração da população inicial e mutação genética, executa-se o método de reparar indivíduos inválidos com o objetivo de manter os indivíduos da população dentro da área factível de busca, ou seja, manter o conjunto de indivíduos como soluções válidas para o problema.

A metodologia está dividida nas seguintes etapas:

- Primeira etapa consistiu em realizar pesquisa bibliográfica do estado da arte sobre Algoritmos Genéticos dos últimos cinco anos;
- Segunda etapa foi representar o PSOJ/D para o AGSOJ/D seleção-mutação baseado em Algoritmo Genético;
- Terceira etapa consistiu em criar um projeto de *software* para Web, chamado AGSOJ/D, utilizando metodologias, padrões, métodos e ferramentas de desenvolvimento de sistemas baseados na Engenharia de Software (SOMMERVILLE, 2007);
- Quarta etapa foi fazer a prototipação do *software* utilizando IDE de desenvolvimento Eclipse para Java;
- Quinta etapa consistiu em representar o PSOJ/D utilizando Simulated Annealing e o AGSOJ/D-X com cruzamento;

- Sexta etapa foi realizar experimentos sobre dois cenários representando as centrais de mandados de Imperatriz e São Luís utilizando as soluções: AGSOJ/D seleção-mutação, AGSOJ/D-X com cruzamento e SA. Os resultados obtidos serviram para criar gráficos com curvas de comparações entre as três abordagens nos dois cenários, além de comparar os resultados do AGSOJ/D seleção-mutação com resultados de processo manual.

Nesse âmbito, este trabalho está dividido da seguinte forma: o Capítulo 2 descreve os conceitos e os componentes em detalhes de Algoritmo Genético e apresenta uma breve introdução ao *Simulated Annealing*; o Capítulo 3 apresenta detalhes do PSOJ/D sobre dois cenários, os quais retratam as centrais de mandados de Imperatriz e São Luís, além de mostrar a quantidade de soluções válidas para cada cenário; o Capítulo 4 mostra detalhes do AGSOJ/D seleção-mutação, como o problema foi modelado e adaptado para solucionar o PSOJ/D e apresenta a representação do PSOJ/D no AGSOJ/D-X com cruzamento e no *Simulated Annealing*; o Capítulo 5 mostra como foram realizados os experimentos nos cenários 1 e 2 apresentando e discutindo os resultados; o Capítulo 6 delineia sobre a conclusão e trabalhos futuros. O projeto de *software* do AGSOJ/D é apresentado no Apêndice A.

2 Meta-Heurísticas

Meta-heurística por definição são métodos heurísticos para resolver de forma genérica problemas de otimização, normalmente da área de otimização combinatória e são geralmente aplicadas a problemas que não se conhece algoritmo eficiente para resolver. O termo meta-heurística, introduzido por Glover (1986) apud (CHAVES, 2009), deriva da composição de duas palavras gregas, heurística (do verbo heuristiké) que significa "arte de encontrar, descobrir", e o prefixo meta (metá) que exprime a ideia de "nível superior, maior generalidade". Este capítulo apresenta duas meta-heurísticas utilizadas como fundamentação teórica desse trabalho: Algorimtos Genéticos e Têmpera Simulada.

2.1 Algoritmos Genéticos

Os algoritmos genéticos vêm sendo aplicados em diversos problemas de otimização desde 1975 com a publicação do livro *Adaptation in Natural and Artificial Systems* de John Holland. Esses algoritmos se baseiam em princípios da teoria da genética populacional, onde uma população aumenta em número através da reprodução e pode ser diversificada pela combinação genética e/ou por mutação. Esta teoria foi construída através da combinação dos estudos de Charles Darwin (darwinismo) através de 20 anos de observações e experimentos que culminou na publicação do livro *On the origin of species by means of natural selection* em 1859 e com o trabalho de Gregor Mendel desenvolvido em 1865 sobre os princípios básicos de herança genética (LINDEN, 2012), (LOBO, 2005).

Algoritmos Genéticos são procedimentos probabilísticos de busca que se baseia na teoria da evolução genética. Os Algoritmos Genéticos trabalham com uma população de soluções que sofrem modificações a cada geração através da aplicação de operadores genéticos. A ideia central é que a solução escolhida seja aquela que tenha melhor adaptação relativa, medida através de um valor de aptidão, que envolve o objetivo global da otimização (REINA, 2012).

Algoritmo Genético é baseado na ideia de que a computação, especialmente o campo da inteligência computacional, tem se beneficiado muito da observação da natureza. Através desta observação surgiram, entre outros, as redes neurais (simulação do comportamento do cérebro humano), a têmpera simulada (simulação do processo de resfriamento de metais) e, finalmente, os Algoritmos Genéticos (simulação da evolução natural) (LINDEN, 2012).

Para Pacheco (1999), Algoritmos Genéticos - AGs (GAs: Genetic Algorithms) são algoritmos matemáticos inspirados nos mecanismos de evolução natural e recombinação genética. A técnica de Algoritmos Genéticos fornece um mecanismo de busca adaptativa

que se baseia no princípio reprodução e sobrevivência dos mais aptos de Darwin. AGs são particularmente aplicados em problemas complexos de otimização: problemas com diversos parâmetros ou características que precisam ser combinadas em busca da melhor solução; problemas com muitas restrições ou condições que não podem ser representadas matematicamente; e problemas com grandes espaços de busca (PACHECO, 1999).

2.1.1 Conceitos inerentes ao AG

A seguir, são apresentadas as principais definições relacionadas aos Algoritmos Genéticos:

- cromossomo ou genótipo: cadeia de caracteres, representando alguma informação relativa às variáveis do problema. Cada cromossomo representa, deste modo, um indivíduo e portanto uma solução do problema;
- gene: é a unidade básica do cromossomo. Cada cromossomo tem certo número de genes, cada um descrevendo certa variável do problema. Podem ser do tipo binário, inteiro ou real, dentre outros tipos;
- alelo: é o valor que o gene por assumir. Em uma representação binária o alelo pode ser 0 (zero) ou 1 (um), somente.
- população: conjunto de cromossomos ou soluções;
- fenótipo: cromossomo decodificado;
- geração: o número da iteração que o algoritmo genético executa, gerando uma nova população a cada iteração;
- operações genéticas: operações que o algoritmo genético realiza sobre cada um dos cromossomos. As mais comuns são cruzamento e mutação;
- região factível ou soluções válidas: o conjunto, espaço ou região que compreende as soluções possíveis ou viáveis do problema a ser otimizado. Deve ser calculada independentemente da representação utilizada para o problema;
- espaço de busca: a área que o algoritmo deve percorrer em busca de boas ou melhores soluções, incluindo as soluções inválidas. É a área total de busca que é altamente dependente da representação utilizada para o problema;
- função objetivo, de aptidão, de avaliação ou *fitness*: construída a partir dos parâmetros envolvidos no problema. Fornece uma medida da proximidade da solução em relação a um conjunto de parâmetros. A função de aptidão permite o cálculo da aptidão de cada indivíduo e fornecerá o valor a ser usado para o cálculo de sua probabilidade de ser selecionado para reprodução;

- aptidão ótima: indivíduo que a possui, é o melhor da população.

A Figura 1 apresenta um diagrama que classifica o AG como um tipo de algoritmo evolucionário dentro de uma técnica de busca aleatória-guiada.

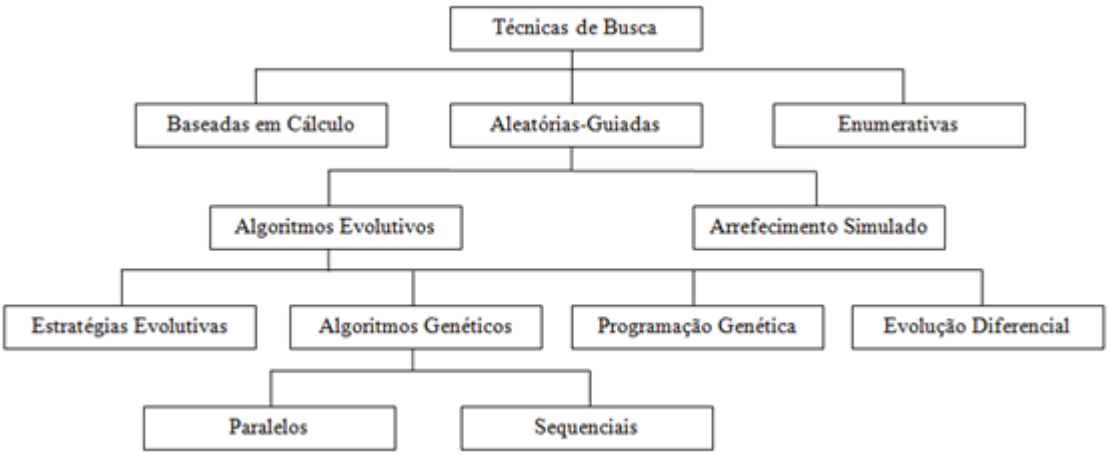

Figura 1 – Técnicas de Busca

Fonte: LINDEN, 2012

De acordo com a Figura 1 acima o Algoritmo Genético é um tipo de Algoritmo Evolucionário - AE. No entanto, os AGs são meta-heurísticas que não asseguram a obtenção do melhor resultado possível em todas as suas execuções. São extremamente dependentes de fatores estocásticos (probabilísticos), tanto na fase de inicialização da população quanto na fase de evolução (durante a seleção dos pais, principalmente). Pode-se dizer que indivíduos com uma melhor adequação do seu fenótipo ao meio ambiente (aptidão melhor) reproduzem mais. Ao reproduzirem mais, têm mais chances de passar seus genes para a próxima geração (LINDEN, 2008), (LINDEN, 2012).

Embora com equipamentos limitados, a partir das décadas de 50 e 60, diversos cientistas começaram a estudar sistemas computacionais que buscavam simular a ideia de evolução de populações. Trata-se do começo do desenvolvimento dos chamados sistemas computacionais evolucionários, com a premissa de que a evolução poderia ser usada como uma ferramenta de otimização para diversos problemas (LOBO, 2005).

Nos Algoritmos Genéticos populações de indivíduos são criados e submetidos aos operadores genéticos: cruzamento, mutação e inver-over. Estes operadores utilizam uma caracterização da qualidade de cada indivíduo como solução do problema em questão chamada de função de aptidão e irão gerar um processo de evolução natural destes indivíduos, que eventualmente deverá gerar um indivíduo que caracterizará uma boa solução (talvez até a melhor possível) para determinado problema (LINDEN, 2012).

Os AGs são algoritmos robustos, genéricos e facilmente adaptáveis, consistem de uma técnica amplamente estudada e utilizada em diversas áreas (CORTES et al.,

2013), (GOMES; GUALDA, 2011), (WANG; ACHARYA; KAM, 2016), sendo seu funcionamento mostrado no Algoritmo 11. Inicialmente uma população de indivíduos é criada usualmente de forma aleatória. Em seguida a população é avaliada. Enquanto o critério de parada não for atingido (normalmente uma quantidade de iterações ou de chamadas à função de aptidão) a população é selecionada para cruzamento. A seleção pode ser feita de várias formas, as mais usadas são: roleta e torneio, mas depende do problema a ser resolvido. Os selecionados são então cruzados de acordo com uma probabilidade de cruzamento até que uma nova população seja criada. Em seguida essa nova população sofre mutação através de uma probabilidade de mutação. Finalmente ocorre o processo de atualização onde a nova população substitui a anterior. No entanto, é possível utilizar-se de elitismo onde o melhor indivíduo é sempre mantido mesmo que ele esteja na população de pais. Detalhes podem ser visto no trabalho de Linden (LINDEN, 2012) e Michalewicz (MICHALEWICZ, 1999).

Algoritmo 1: Algoritmo Genético - AG

```
   Entrada: AG Parâmetros
1  inicio
2      Pop = Inicializar(x)
3      Avalia(Pop)
4      while critério de parada não for atingido do
5          Cruzamento(Pop)
6          Mutação(Pop)
7          Avalia(Pop)
8          Pop = Atualização(Pop)
9      end
10 fin
11 return Pop
```

Cada componente de AG apresentado nesse algoritmo representa um método peculiar que deve ser implementado de acordo com a representação do problema na meta-heurística de Algoritmo Genético com o objetivo de tentar obter as melhores soluções dentro do espaço de busca do problema.

A seguir será apresentado a principal etapa antes da implementação propriamente dita do Algoritmo Genético, que é a representação e também a descrição detalhada de cada um desses componentes.

2.1.2 Representação

Nesta etapa é onde deve ser criado um link entre o "mundo real"e o "mundo AG", ou seja, definir a ponte entre o problema original e o espaço de busca a ser explorado pelo Algoritmo Genético. Definir uma forma de representação adequada é uma tarefa bastante difícil e pode ser decisiva para o sucesso do Algoritmo Genético. Geralmente, uma

boa codificação vem com bastante prática e bom conhecimento do domínio de aplicação MITCHELL, (1998) apud (BASGALUPP, 2010), (LINDEN, 2012), (LOBO, 2005).

A representação do cromossomo que representa um indivíduo é fundamental para o Algoritmo Genético. Ela consiste em uma maneira de traduzir a informação do nosso problema em uma maneira viável de ser tratada pelo computador. Quanto mais ela for adequada ao problema, maior a qualidade dos resultados obtidos. Cada pedaço indivisível desta representação é chamado de um gene, por analogia com as partes fundamentais que compõem um cromossomo biológico. E importante notar que a representação cromossômica é completamente arbitraria, ficando sua definição de acordo com a adequação ao problema. As principais regras de uma boa representação segundo (LINDEN, 2012) são:

- A representação deve ser a mais simples possível;
- Se houver soluções proibidas ao problema, então é preferível que elas não tenham uma representação;
- Se o problema impuser condições de algum tipo, estas devem estar implícitas dentro da representação.

A forma de representação é característica de cada problema em questão. A representação binária é a mais conhecida, sendo muito usada em AGs. As formas mais comuns de representação são apresentadas a seguir.

2.1.2.1 Representação binária

A representação binária é a mais simples e mais usada pelos praticantes da área dos Algoritmos Genéticos. Um cromossomo nada mais é do que uma sequência de bits. Cada gene é somente um *bit*. O conceito representado por cada *bit* e/ou conjunto de *bits* é inerente ao problema (LINDEN, 2012).

A codificação binária é uma forma de codificação bastante utilizada na literatura por diversas razões. A principal delas é o fato de diversas teorias de convergência dos AGs serem baseadas na pressuposição de que a forma de representação é essa. Além disso, operações como cruzamento e mutação são mais fáceis de serem aplicadas quando se utiliza esse tipo de codificação. No entanto, a desvantagem aparece quando o problema de otimização é muito complexo, por exemplo, sendo necessário *strings* muito longas (BASGALUPP, 2010). A Figura 2 ilustra um exemplo de indivíduo nesse formato, em que pode ser observado que a *string* é composta por bits (0's e l's).

A Figura 2 apresenta um indivíduo com representação binária, cujo cromossomo possui tamanho 10.

Figura 2 – Indivíduo codificado com uma *string* de binários

2.1.2.2 Representação inteira

Em muitas situações, a representação binária não é a mais adequada para codificar a solução de um problema. Um exemplo típico seria o problema de encontrar os valores ótimos para um conjunto de variáveis cujos domínios sejam os inteiros. Seria muito mais natural representar a solução como um *string* de números inteiros, sendo o tamanho do *string* igual ao número de variáveis e cada posição (gene do indivíduo) representando o valor da variável em questão, (BASGALUPP, 2010). A Figura 3 ilustra um possível indivíduo para a solução do problema exemplificado.

Figura 3 – Indivíduo codificado com uma *string* de inteiros

A Figura 3 apresenta um indivíduo com representação inteira, cujo cromossomo possui tamanho 9.

2.1.2.3 Representação por valores reais

Pela mesma razão da representação anterior, em diversas aplicações é mais natural utilizar valores reais para formar os indivíduos. Um exemplo bem típico deste tipo de codificação é quando se quer otimizar os pesos de uma rede neural, (BASGALUPP, 2010). Este tipo de representação pode ser visualizado na Figura 4.

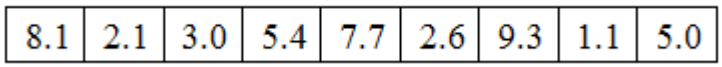

Figura 4 – Indivíduo codificado com uma *string* de valores em reais

A Figura 4 apresenta um indivíduo com representação por valores reais, cujo cromossomo possui tamanho 9.

Os tópicos a seguir descrevem com mais detalhes cada um dos componentes do Algoritmo Genético 1 apresentado na Figura **??**.

2.1.3 População inicial

A geração da população inicial é o processo no qual são criados os primeiros indivíduos. O número de indivíduos da população inicial é um parâmetro fornecido pelo

usuário. É desejável que os indivíduos sejam os mais diferentes possíveis entre si, de modo que estejam espalhados no espaço de busca das soluções para o problema em questão, aumentando, destarte, a diversidade da população. O método mais comum de geração de população inicial é o método no qual os indivíduos têm seus genes preenchidos com valores aleatórios. Supondo que o cromossomo seja uma *string* binária, um exemplo de preenchimento aleatório dos genes é assumir uma probabilidade p para cada gene receber o valor 1 e $(1 — p)$ para receber o valor 0. Espera-se que os indivíduos que compõem a população inicial representem soluções factíveis para o problema em questão, (BASGALUPP, 2010).

Para garantir a factibilidade dessas soluções utiliza-se neste ponto conhecimento sobre o problema. Um exemplo desse conhecimento é o domínio dos genes preenchidos em um *string* com valores reais, pois dependendo do problema de otimização, não é qualquer valor que pode ser atribuído a qualquer gene do indivíduo.

2.1.4 Função de Aptidão

Nesta etapa, é medido o grau de aptidão dos indivíduos da população utilizando a função de aptidão. A função de aptidão é, junto com a etapa de representação, um dos componentes do Algoritmo Genético que está diretamente relacionada com o domínio do problema em questão. Na função de aptidão, um ou mais critérios (objetivos) podem ser utilizados para otimização, (BASGALUPP, 2010). Este é o componente mais importante de qualquer Algoritmo Genético. É através dessa função que se mede o quanto o indivíduo está próximo da solução desejada ou pelo menos próximo de um ótimo local (LOBO, 2005).

Uma função de aptidão também penaliza soluções implausíveis para o problema e avalia satisfatoriamente o grau de adequação de cada indivíduo como solução do problema em questão, de forma que uma solução que confira maior benefício receba uma aptidão mais alta do que aquela que confere um benefício menor. A função de aptidão deve ser escolhida com grande cuidado. Ela deve agregar todo o conhecimento que se possui do problema a ser resolvido, tanto as restrições quanto seus objetivos de qualidade (LINDEN, 2012). A Figura 5 ilustra a representação dos valores de aptidão (avaliação, também do inglês *fitness*) para cada indivíduo da população.

A Figura 5 apresenta os valores de x e y em inteiros apenas por questão de simplicidade, pois a representação desse exemplo é binária, conforme está demonstrado na equação 2.1.

$$f(x, y) = |x * y * sen(\frac{y}{4})| \tag{2.1}$$

A equação 2.1 apresenta uma função multimodal, contendo vários máximos, sendo que ambas as variáveis (x e y) pertencem ao intervalo dado por $[-100; 100]$.

Nº Ordem no Vector	*x*	*y*	*Avaliação*
0	3	1	3,1
1	5	9	32,8
2	2	16	1,0
3	4	5	15,1

Figura 5 – Representação dos valores de aptidão

Fonte: LINDEN, 2012

2.1.5 Elitismo

O elitismo foi introduzido por Kenneth De Jong (JONG, 1975 apud BASGALUPP, 2010) e pode ser utilizado por vários métodos de seleção. Essa propriedade garante que um número x de melhores indivíduos seja mantido na próxima geração da população. Esses melhores indivíduos poderiam ser perdidos após as operações de cruzamento e mutação. Segundo Mitchell (1998) apud BASGALUPP (2010), muitos pesquisadores têm encontrado uma melhoria significativa na performance dos AGs quando utilizam o elitismo. Entretanto, é bom tomar bastante cuidado com o valor de x, pois se muito grande pode causar o domínio prematuro dos indivíduos mais aptos desde as gerações iniciais, implicando em uma rápida perda de diversidade da população.

2.1.6 Seleção

É no estágio de seleção que são escolhidos os indivíduos que serão submetidos ao cruzamento. Nesse ponto, faz-se uso do grau de aptidão aptidão dos indivíduos, e os mais aptos possuem probabilidade maior de serem escolhidos como reprodutores em alguns métodos de seleção, por exemplo a roleta. É importante destacar que a escolha é probabilística, não sendo garantido, portanto, que o melhor indivíduo (o mais apto) será um dos selecionados para formar a próxima geração da população (BASGALUPP, 2010), (LINDEN, 2012), (LOBO, 2005).

Os métodos de seleção mais comuns, são apresentados a seguir:

2.1.6.1 Roleta

Neste método de seleção, cada indivíduo, representados pelas letras de a a j, possui um intervalo de números de uma roleta que pode ser melhor observado na Figura 6. Esse intervalo de números é proporcional ao valor de aptidão do indivíduo. A roleta é girada n vezes, em que n é o número de pais que serão selecionados. A cada giro da roleta, um número z é sorteado indicando uma fração da roleta, sendo que cada número da roleta pode ser sorteado de acordo com sua aptidão. O indivíduo selecionado é aquele que possuir

o número z dentro de seu intervalo de números. No final do processo, são selecionados n indivíduos que poderão participar do cruzamento (BASGALUPP, 2010), (LINDEN, 2012), (LOBO, 2005).

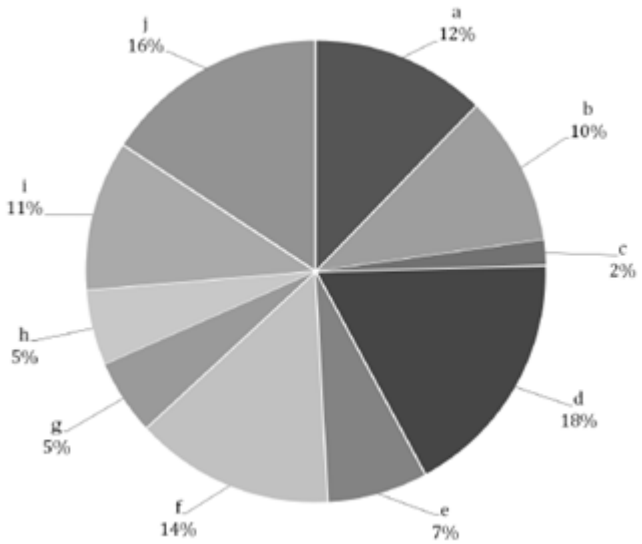

Figura 6 – Seleção por roleta

Fonte: BASGALUPP, 2010

Ao observar a Figura 6 pode-se entender melhor a seleção por roleta. Cada indivíduo tem seu espaço na roleta proporcional a sua aptidão. Logo o indivíduo d, com aptidão de 18%, tem a maior probabilidade de ser selecionado.

2.1.6.2 Ranking

O método *ranking* pode prevenir a convergência prematura para ótimos locais no espaço de soluções do problema. Nesse caso, os indivíduos da população são ordenados de acordo com a sua aptidão e a probabilidade de escolha é atribuída conforme a posição do indivíduo no *ranking*. Quando se utiliza esse método, o problema de grandes diferenças entre as chances de seleção dos indivíduos é eliminado, uma vez que a relação das probabilidades de escolha entre dois indivíduos de posições i e $i + 1$ no ranking é independente das diferenças absolutas de seus valores de aptidão, dependendo apenas das posições no *ranking* (BASGALUPP, 2010).

O *ranking* evita dar probabilidade maior de seleção para um grupo pequeno de indivíduos altamente aptos, reduzindo, assim, a pressão da seleção quando a variação da aptidão é muito elevada.

2.1.6.3 Torneio

De acordo com (LINDEN, 2012), o método do torneio, como o próprio nome diz, consiste em selecionar uma série de indivíduos da população e fazer com que eles entrem em competição direta pelo direito de ser pai, usando como arma a sua aptidão.

Neste método, existe um parâmetro denominado tamanho do torneio k que define quantos indivíduos são selecionados aleatoriamente dentro da população para competir. Uma vez definidos os competidores, aquele dentre eles que possui a melhor aptidão é selecionado para a aplicação do operador genético.

O valor mínimo de k é igual a 2, pois do contrário não haverá competição, mas não há nenhum limite teórico para o valor máximo deste parâmetro. Se for escolhido o valor igual ao tamanho da população n o vencedor será sempre o mesmo (o melhor de todos os indivíduos) e se forem escolhidos valores muito altos (próximos do tamanho da população), os n-k indivíduos tenderão a predominar, uma vez que sempre um deles será o vencedor do torneio.

Os indivíduos são selecionados para participar do torneio de forma completamente aleatória. Não existe nenhum favorecimento para os melhores indivíduos, como no caso da roleta. A única vantagem que os melhores indivíduos da população têm é que, se selecionados, eles vencerão o torneio (LINDEN, 2012).

Após uma rodada do torneio, todos os participantes, inclusive o vencedor, retornam à população original e podem novamente ser selecionados para os próximos torneios. Esse processo se repete até que o número desejado de reprodutores seja alcançado. A seleção por torneio é, provavelmente, a mais utilizada na literatura de Algoritmos Genéticos. Isso se explica, principalmente, por sua extrema simplicidade e pelo fato de que a pressão seletiva pode ser facilmente controlada ao variar o tamanho t do torneio. Outra vantagem deste método é que, assim como o ranking, não há a necessidade de normalizar a aptidão dos indivíduos, visto que o fato de um indivíduo ser duas ou três vezes mais apto que outro não é o que determina a decisão da escolha, mas sim qual deles é o melhor, (BASGALUPP, 2010).

2.1.6.4 Truncada

Ricardo Linden (2012) explica o uso da seleção truncada, onde apenas os melhores $x\%$ da população poderão ser escolhidos como país da próxima geração. O valor x é um parâmetro do algoritmo, que pode variar de 1% a 100%. Os valores mais usuais para x são aqueles na faixa [l0%-50%]. Para implementar este método, os indivíduos são ordenados de forma decrescente de acordo com sua aptidão e somente aqueles cujas posições estiverem entre 1 e a posição de corte poderão participar da seleção. A necessidade desta ordenação faz com que o algoritmo tenha uma complexidade mínima de tempo de O($nlogn$) por

rodada. Qualquer outro método citado anteriormente pode ser combinado com a seleção truncada.

Esse método, segundo demonstrou Blicke (1997) apud Linden (2012), é o que causa maior perda de diversidade e por isso é o que tende a fazer com que o AG apresente o pior desempenho. Porém nada impede desse método ser utilizado como uma solução híbrida para o AG.

2.1.7 Operadores genéticos

Os operadores genéticos consistem em aproximações computacionais de fenômenos vistos na natureza, como a reprodução sexuada, a mutação genética e quaisquer outros que a imaginação dos programadores consiga reproduzir (LINDEN, 2012). A seguir são apresentados os principais operadores genéticos.

2.1.7.1 Cruzamento

Após serem selecionados os pares de indivíduos (pais), eles podem se reproduzir. Cada par de pais pode gerar um par de filhos, dependendo do método. Cada filho é composto por uma combinação dos genes dos pais. Quais genes dos pais passarão para cada um dos filhos é determinado pelo método de cruzamento utilizado. Cabe salientar que o cruzamento acontece de acordo com uma probabilidade, chamada de probabilidade do cruzamento, que em geral é alta, que é um parâmetro pré-determinado pelo usuário. Caso não ocorra o cruzamento, os filhos são exatamente cópias de seus pais, (BASGALUPP, 2010). Os tipos mais comuns de cruzamento nos AG's binários são:

- Cruzamento de 1 ponto

  Este tipo de cruzamento é composto por um ponto de cruzamento em uma posição p, em que o filho l recebe os genes do pai até o ponto p, e os genes da mãe do ponto $p + 1$ em diante. Já o filho 2 recebe os genes da mãe até o ponto p e os genes do pai do ponto $p + 1$ em diante. Esse processo pode ser visualizado na Figura 7, na qual os dois indivíduos de cima representam os pais e os dois abaixo representam os filhos, (BASGALUPP, 2010).

  A Figura 7 apresenta o cruzamento de 1 ponto. O corte na posição 5 indica que os genes 1 ao 4 do indivíduo 1 (0110) combinado com os genes 5 ao 9 do indivíduo 2 (00011) formarão o filho 1 (011000011). Os genes 5 ao 9 do indivíduo 1 (01110) combinados com os genes 1 ao 4 do indivíduo 2 (1001) formarão o filho 2 (100101110).

- Cruzamento multi-pontos

  Esse cruzamento é semelhante ao cruzamento de 1 ponto, porém pode haver n pontos de cruzamento. A Figura 8 ilustra os filhos gerados para um exemplo com $n = 2$.

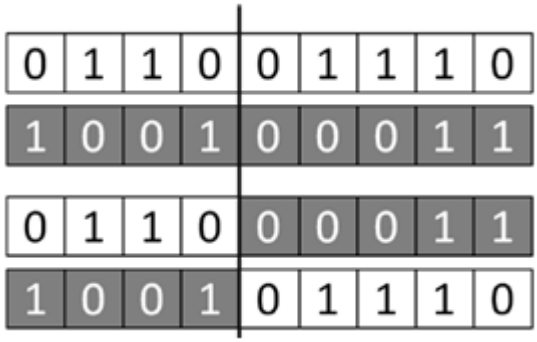

Figura 7 – Cruzamento de 1 ponto

Fonte: BASGALUPP, 2010

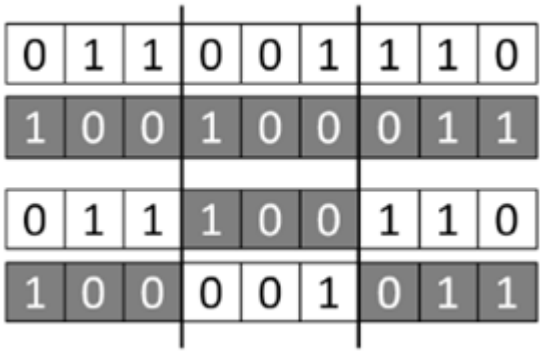

Figura 8 – Cruzamento multi-pontos

Fonte: BASGALUPP, 2010

A Figura 8 apresenta o cruzamento de múltiplos pontos. Foram realizados 2 cortes nas posições 4 e 7. Nesse tipo de cruzamento os genes são combinados de forma intercaladas, ou seja, genes do indivíduo 1 + genes do indivíduo 2 + genes do indivíduo 1 formarão o filho 1 e genes do indivíduo 2 + genes do indivíduo 1 + genes do indivíduo 2 formarão o filho 2.

- Cruzamento uniforme

  No cruzamento uniforme, os genes a serem trocados são escolhidos aleatoriamente em cada execução. A Figura 9 ilustra o exemplo de uma execução de cruzamento uniforme.

  O cruzamento uniforme apresentado na Figura 9 combina os genes escolhido aleatoriamente do indivíduo 1 com genes da mesma posição do indivíduo 2 para formar o filho 1. Os genes não selecionados do indivíduo 1 são combinados com os genes na mesma posição do indivíduo 2 para forma o filho 2.

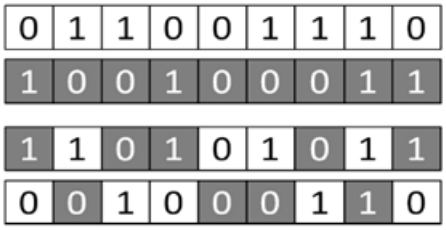

Figura 9 – Cruzamento uniforme

Fonte: BASGALUPP, 2010

2.1.7.2 Mutação

Após todo o processo de cruzamento, antes da nova população ser definida, os indivíduos passam pelo processo de mutação. A mutação é uma alteração aleatória no valor dos genes de um indivíduo e ocorre com uma probabilidade pré-determinada pelo usuário, chamada de probabilidade de mutação, que em geral é baixa. Pode-se efetuar a mutação de um gene trocando seu valor por outro qualquer, somando ou subtraindo um valor aleatoriamente escolhido, trocando os valores dos genes de posições aleatórias, dentre outras opções. A mutação destina-se a especular novos pontos no espaço de busca de soluções, explorando novas possibilidades, como também a de evitar que aconteça a chamada "convergência prematura", evento no qual o algoritmo converge para um ótimo local no espaço de soluções do problema, paralisando o processo evolutivo, (BASGALUPP, 2010). A Figura 10 ilustra uma possível mutação para um indivíduo com representação binária, em que quatro de seus genes são alterados.

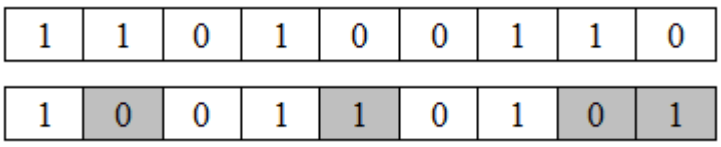

Figura 10 – Exemplo de mutação com representação binária

A Figura 10 apresenta um exemplo de mutação em um indivíduo com representação binária. Os genes que sofreram mutação estão em destaque no novo indivíduo.

2.1.7.3 Inver-Over

O operador *Inver-Over* trabalha de forma isolada, substituindo os operadores de cruzamento e mutação. Esse operador aplica uma forte seleção sobre os indivíduos e usa um ou dois pais para cada iteração (LINDEN, 2012). O resto do AG procede como todo AG tradicional. A única exceção é o fato de que o operador *Inver-Over* é aplicado a todos os pais, ao invés de uma seleção de pais feita por algum mecanismo aleatório, como a roleta. Isto ocorre, pois o operador *Inver-Over* é equivalente, de certa forma, a um processo de

busca local, otimizando as conexões localmente para melhorar o indivíduo corrente ao máximo (LINDEN, 2012).

Ricardo Linden cita Michalewicz (2010) sobre uma série de estudos realizados, os quais mostram que os AGs que usam o operador Inver-Over são extremamente velozes e precisos, tendo atingido um resultado muito próximo da solução ótima em vários casos de teste. Como sempre, não há um estudo teórico formal que prove que esta situação se repetirá em todos os problemas possíveis.

Um estudo aprofundado sobre a meta-heurística de AG foi necessário para modelar o PSOJ/D como um problema de otimização e aplicar os recursos de busca aleatória-guiada do AG para encontrar as melhores soluções dentro do espaço de busca dos dois problemas citados na Seção 4.1.1. A seguir será apresentado uma breve introdução à meta-heurística de Têmpera Simulada.

2.2 Têmpera Simulada

O algoritmo Têmpera Simulada, do inglês *Simulated Annealing*, proposto por Kirkpatrick (KIRKPATRICK; GELATT; VECCHI, 1983) foi inspirado pelo processo de arrefecimento (annealing) de sistemas físicos, um procedimento utilizado para levar um material a seu estado de equilíbrio máximo, ou seja, a um estado de energia mínima. Mais detalhes sobre o uso dessa meta-heurística podem ser vistos nos trabalhos de (HU et al., 2015) e (SOUSA; CUNHA; MARQUES, 2006). O Algoritmo 2, demonstra o pseudo-código do SA. O algoritmo inicia-se criando uma solução x aleatoriamente na região factível do problema. Em seguida é calculado o valor energético da solução x. Depois inicia-se o processo de têmpera simulada, laço que se inicia na linha 3 até linha 14. O método é terminado quando nenhuma melhora significativa for alcançada, um número fixo de iterações for atingido, ou a temperatura T atingir seu valor mínimo. Na linha 4 é feita uma cópia da solução x para x' que em seguida é modificada aleatoriamente e o calculado o valor energético de x'. Se valor energético x' for melhor que o valor x, então x' é copiado para x, senão a probabilidade de x' ser aceito como o ponto atual é dada por um caso particular da distribuição de Boltzmann-Gibbs, linhas 8 a 11. Isso permite o algoritmo sair de ótimos locais, já que admite probabilisticamente soluções piores. Depois na linha 13 é reduzido a temperatura e o ciclo de têmpera simulada inicia-se novamente até que o critério de parada seja atendido.

Não houve um estudo aprofundado sobre a meta-heurística de SA, pois o objetivo foi apenas utilizar uma outra abordagem para comparar os resultados obtidos com a solução principal proposta utilizando a meta-heurística de AG.

A seguir será apresentado detalhes e tamanho do PSOJ/D e porque utilizar a meta-heurística de AG para resolvê-lo, bem como restrições e penalidades inerentes ao

Algoritmo 2: Simulated Annealing - SA

```
   Entrada: SA Parâmetros
 1 inicio
 2 |   x = Criar solucao(x)
 3 |   while temperatura > temperatura mínima do
 4 |   |   x' = vizinho(x)
 5 |   |   if valor energético (x') > valor energético (x) then
 6 |   |   |   x = x'
 7 |   |   else
 8 |   |   |   D = valor energético (x') - valor energético (x)
 9 |   |   |   if exp(-D / temperatura) > Aleatório(1) then
10 |   |   |   |   x = x'
11 |   |   |   end
12 |   |   end
13 |   |   Reduz temperatura
14 |   end
15 fin
16 return x
```

PSOJ/D.

2.3 Considerações finais

Este capítulo apresentou as duas meta-heurísticas aplicadas ao PSOJ/D. A Seção 2.1, sobre Algoritmos Genético, descreveu os conceitos inerentes ao AG e suas definições; em seguida foram apresentadas as principais representações de um AG e depois mostradas em detalhes cada componente de um Algoritmo Genético canônico. A Seção 2.2 apresentou uma breve introdução ao *Simulated Annealing*.

3 Problema PSOJ/D

O PSOJ/D é um problema regido pelo Provimento 18/2011 de 19 de agosto de 2011 da Corregedoria Geral de Justiça considerando o disposto na Lei Complementar nº 85, de 21 de junho de 2005, que criou a Central de Mandados da Comarca de São Luís e Imperatriz. O Provimento 18/2011, Art. 2º, disciplina que a alocação dos oficiais de justiça nas respectivas áreas (distritos) dar-se-á por sorteio (distribuição), realizado eletronicamente pela TI - Tecnologia da Informação (atualmente Divisão de Sistemas de Informação) do Tribunal de Justiça (PROVIMENTO 18, 2011). Desde da criação da Central de Mandados em 2005 a distribuição dos oficiais de justiça é feito manualmente ou com auxílio de planilha do Excel.

3.1 Conceitos inerentes ao PSOJ/D

A seguir são apresentados alguns conceitos que são inerentes ao PSOJ/D e que visa uma melhor compreensão do mesmo.

- **Comarca**

  É termo jurídico que designa uma divisão territorial específica, que indica os limites territoriais da competência de um determinado juiz ou juízo de primeira instância.

- **Central de Mandado**

  É uma secretaria física constituída com o propósito de centralizar a gestão de entrega de mandados por oficiais de justiça nos distritos. Em uma central de mandados, cada Oficial de Justiça tem sua região de cobertura, com foco no destinatário do mandado, desvinculando o mesmo das varas e ofícios de origem do mandado. Assim, os oficiais de justiça cumprem as suas diligências com muito mais rapidez, considerando que os endereços pertencem à mesma região, sendo perto uns dos outros (FREITAS, 2016).

- **Distrito**

  É constituído de um ou mais bairros adjacentes de acordo com a região geográfica de cada município com a finalidade de agilizar a logística de entrega de mandados por oficiais de justiça.

- **Oficial de Justiça**

  É o profissional do poder judiciário lotado em distrito ou secretaria (quando a comarca não tem central de mandado) responsáveis por entregar mandados às partes do processo. É comum se dizer, no âmbito jurídico, que o Oficial de Justiça é a longa

manus do Magistrado, ou seja, as mãos estendida do Juiz na rua. Isso porque é ele quem executa, de forma efetiva e material, as determinações que o Juiz registra no papel (FREITAS, 2016).

A Figura 11 demonstra como esses conceitos estão relacionados.

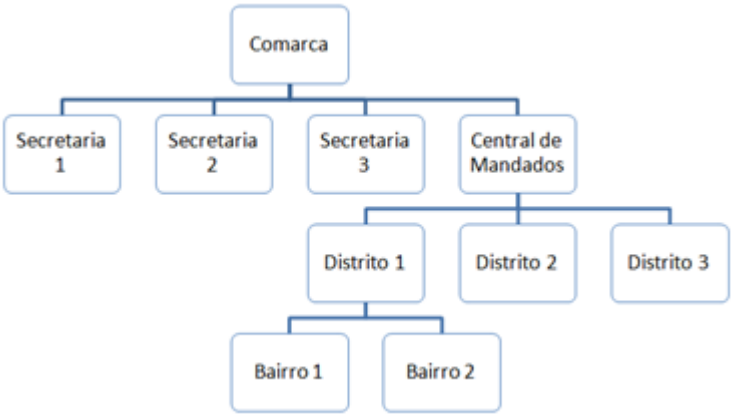

Figura 11 – Conceitos inerentes ao PSOJ/D.

Pode-se observar na Figura 11 apresentada acima, que a comarca é constituída de secretarias e a central de mandados é uma secretaria que é composta por distritos e os distritos são constituídos por bairros.

- **Distribuição**

  Os usuários das centrais de mandados chamam de sorteio. É o nome dado à atividade de selecionar os oficiais de justiça por distritos, trimestralmente, com o objetivo de manter o rodízio desses oficiais de justiça entre os distritos em atendimento ao Provimento 18/2011 da CGJ-MA.

3.2 Problema da central de mandados de Imperatriz

A central de mandado de Imperatriz é composta de 27 oficiais de justiça que devem ser distribuídos entre os 6 distritos existentes a cada três meses na forma de rodízio. A Tabela 1 mostra a composição da Central de Mandados de Imperatriz com suas respectivas capacidades de oficiais de justiça por distrito.

A Tabela 1 apresentada mostra que a quantidade de soluções válidas possíveis é calculada pelo produto da combinação de 27 oficiais de justiça entre 6 distritos com diferentes capacidades. Onde cada distrito é combinado da seguinte maneira:

- Distrito 1 que é o primeiro, combina com os 27 oficiais de justiça tomados 13 a 13;

Tabela 1 – Problema da central de mandado de Imperatriz

Distrito	Capacidade de OJ	Soluções válidas
Distrito 1	13	Combin(27, 13) = 20.058.300
Distrito 2	3	Combin(14, 3) = 364
Distrito 3	3	Combin(11, 3) = 165
Distrito 4	3	Combin(8, 3) = 56
Distrito 5	3	Combin(5, 3) = 10
Plantão	2	Combin(2, 2) = 1
Total: 6	27	$\approx 6,70 * 10^{15}$

Legenda: OJ = Oficial de Justiça

- Distrito 2 combina com 14 $(27 - 13 = 14)$ oficiais de justiça tomados 3 a 3, pois 13 dos 27 oficiais de justiça já foram combinados no distrito 1;
- Distrito 3 combina com 11 $(14 - 3 = 11)$ oficiais de justiça tomados 3 a 3, pois 3 dos 14 oficiais de justiça já foram combinados no distrito 2;
- Distrito 4 combina com 8 $(11 - 3 = 8)$ oficiais de justiça tomados 3 a 3, pois 3 dos 11 oficiais de justiça já foram combinados no distrito 3;
- Distrito 5 combina com 5 $(5 - 3 = 2)$ oficiais de justiça tomados 3 a 3, pois 3 dos 8 oficiais de justiça já foram combinados no distrito 4;
- E finalmente, o Plantão combina com os últimos 2 dois oficiais de justiça tomados 2 a 2.

Nota-se que a partir do distrito 2 são subtraídos os oficiais de justiça combinados em distritos anteriores. O produto das combinações dos seis distritos representa a quantidade de soluções possíveis desse problema. Contudo, o total de $\approx 6,70 * 10^{15}$ não representa o espaço de busca do problema, pois esse depende da representação do problema em AG. A Seção 4.1.1 demonstra a representação em binário dos cenários 1 e 2 e seus respectivos espaços de buscas.

O total de combinações possíveis demostrado na Tabela 1 também pode ser calculado pela seguinte fórmula de análise combinatória: $\frac{27!}{(13!*3!*3!*3!*3!*2!)}$, onde o dividendo, 27!, representa o total de oficiais de justiça e o divisor $13! * 3! * 3! * 3! * 3! * 2!$ representa as capacidades de oficiais de justiça dos distritos 1, 2, 3, 4, 5 e plantão, respectivamente apresentados na Tabela 1. Em outras palavras, o total de combinações representa a quantidade de maneiras que pode-se selecionar os 27 oficiais de justiça por 6 distritos diferentes e cada distrito selecionando n oficiais de justiça.

Atualmente, a distribuição da central de mandado de Imperatriz é feita manualmente executando os seguintes passos:

1. Colocar no plantão os 2 oficiais de justiça indicados pela chefe da central de mandado;

2. Colocar no distrito 2 aqueles que estavam na última distribuição no distrito 1 e que a mais tempo não tinham sido do distrito 2;

3. Colocar no distrito 3 aqueles que estavam na última distribuição no distrito 1 e que a mais tempo não tinham sido do distrito 3;

4. Colocar no distrito 4 aqueles que estavam na última distribuição no distrito 1 e que a mais tempo não tinham sido do distrito 4;

5. Colocar no distrito 5 aqueles que estavam na última distribuição no distrito 1 e que a mais tempo não tinham sido do distrito 5;

    a) Se algum dos distritos 2, 3, 4 ou 5 ficou com mais de 3 oficiais de justiça, então faça a troca (considerando qual oficial está a mais tempo sem ir no distrito em questão);

6. Colocar no distrito 1, os 13 oficiais de justiça restantes. Após realizar o procedimento acima mencionado é criado um *script* para incluir na base de dados o resultado referente ao trimestre.

Esse método de distribuição é aplicado muito bem para a Central de Mandados de Imperatriz, porque quase 50% dos oficiais de justiça são selecionados para o distrito 1. Porém, a Central de Mandados de São Luís que é composta por aproximadamente 84 oficiais de justiça com 16 distritos e cada distrito com quantidades variáveis de oficiais de justiça, esse método é inviável.

3.3 Problema da central de mandados de São Luís

A distribuição na Central de Mandados de São Luís é feita utilizando duas planilhas do Excel, sendo que a primeira planilha serve para distribuir todos os 84 oficiais justiça, criando uma lista que serve para determinar a ordem em que estes serão alocados nos distritos. A segunda planilha contém o histórico dos últimos distritos que cada oficial de justiça já fora alocado, então cada oficial de justiça da primeira planilha na ordem em que foram sorteados é alocado no distrito onde o mesmo ainda não foi distribuído ou no distrito mais antigo que já foi distribuído até o limite da capacidade de cada distrito. A Tabela 2 demonstra o problema da Central de Mandados de São Luís.

A Tabela 2 apresentada representa o problema da Central de Mandados de São Luís com 16 distritos e 84 oficiais de justiça. Assim como o problema da Central de Mandados de Imperatriz, a quantidade de soluções válidas é calculado pelo produto da combinação de 84 oficiais de justiça entre 16 distritos com diferentes capacidades cada distrito.

Tabela 2 – Problema da central de mandados de São Luís

Distrito	Capacidade de OJ	Soluções válidas
Distrito 1	3	Combin(84, 3) = 95284
Distrito 2	5	Combin(81, 5) = 25621596
Distrito 3	5	Combin(76, 5) = 18474840
Distrito 4	7	Combin(71, 7) = 1329890705
Distrito 5	7	Combin(64, 7) = 621216192
Distrito 6	5	Combin(57, 5) = 4187106
Distrito 7	5	Combin(52, 5) = 2598960
Distrito 8	6	Combin(47, 6) = 10737573
Distrito 9	3	Combin(41, 3) = 10660
Distrito 10	10	Combin(38, 10) = 472733756
Distrito 11	8	Combin(28, 8) = 3108105
Distrito 12	4	Combin(20, 4) = 4845
Distrito 13	3	Combin(16, 3) = 560
Distrito 14	6	Combin(13, 6) = 1716
Distrito 15	4	Combin(7, 4) = 35
Distrito 16	3	Combin(3, 3) = 1
Total:	84	$\approx 1,10 * 10^{89}$

Legenda: OJ = Oficial de Justiça

Nota-se que a partir do distrito 2 são subtraídos os oficiais de justiça combinados em distritos anteriores. O produto das combinações dos dezesseis distritos representa o total de combinações possíveis desse problema.

O total de combinações possíveis demostrado na Tabela 2 pode também ser calculado pela seguinte fórmula de análise combinatória: $\frac{84!}{(3!5!5!7!7!5!5!6!3!10!8!4!3!6!4!3!)}$, onde o dividendo, 84!, representa o total de oficiais de justiça e o divisor (3!5!5!7!7!5!5!6!3!10!8!4!3!6!4!3!) representa as capacidades de oficiais de justiça dos distritos 1 ao 16, respectivamente apresentados na Tabela 2. Em outras palavras, o total de combinações representa a quantidade de maneiras que pode-se selecionar os 84 oficiais de justiça por 16 distritos diferentes e cada distrito selecionando n oficiais de justiça.

Logo, as soluções que apresentam distritos com quantidades de oficiais de justiça menor ou maior que suas respectivas capacidades são consideradas soluções inválidas para o PSOJ/D. Por exemplo, o distrito 11 apresentado na Tabela 2 deve selecionar exatamente 8 oficiais de justiça, caso contrário, a solução é inválida, mesmo se os demais distritos selecionarem corretamente seus oficiais de justiça.

O objetivo primordial do algoritmo é alocar cada Oficial de Justiça no melhor distrito. Trata-se pois de um problema em que a quantidade de soluções possíveis ultrapassa a capacidade humana de raciocínio para uma seleção manual.

Apresentado as instâncias e as soluções válidas do PSOJ/D nos dois cenários, percebe-se que é um problema NP-Completo, o que justifica sua resolução através da

utilização de uma meta-heurística baseada em AG. A seguir são apresentados as restrições e penalidades do PSOJ/D.

3.4 Retrições e penalidades

A primeira restrição imposta pelo Provimento 18/2011 da CGJ-MA é que o oficial de justiça não deve ser selecionado para o mesmo distrito da distribuição anterior. A penalidade para essa violação será computada negativamente na qualidade da solução completa.

A segunda restrição é que todos os oficiais de justiça que participam da distribuição devem ser selecionados e todos os distritos selecionar seus respectivos oficiais de justiça de acordo com suas capacidades. Para essa restrição a penalidade é que a solução é inviável para o PSOJ/D. Para evitar essa anomalia é executado um algoritmo construtivo de reparo que faz a correção necessária, selecionando uma solução válida para o PSOJ/D.

Essas restrições são aplicáveis às três meta-heurísticas propostas ao PSOJ/D, apresentadas a seguir.

3.5 Considerações finais

Este capítulo apresentou os conceitos inerentes ao PSOJ/D, em seguida os detalhes do PSOJ/D sobre dois cenários, os quais retratam as centrais de mandados de Imperatriz e São Luís, além de mostrar a quantidade de soluções válidas para cada cenário e depois descreveu as duas restrições do PSOJ/D e as penalidades previstas, caso essas restrições sejam violadas.

4 Meta-Heurística Proposta

A primeira solução proposta é uma meta-heurística nova baseada em algoritmo genético chamada AGSOJ/D seleção-mutação, logo, essa solução não é um algoritmo genético canônico como o apresentado na Seção 2.1. Contudo, foi implementado o AG canônico chamado AGSOJ/D-X, com operador de cruzamento, para comparação com o AGSOJ/D seleção-mutação. Além disso, também, foi implementado o algoritmo de Têmpera Simulada, completando as três soluções desenvolvidas para o PSOJ/D.

4.1 AGSOJ/D seleção-mutação

O Algoritmo Genético para Seleção de Oficiais de Justiça por Distritos (AGSOJ/D) é um algoritmo de seleção-mutação genética para resolver o PSOJ/D. Trata-se de um novo algoritmo baseado em AG. Existem três diferenciais importantes nesse algoritmo: o primeiro, está no método peculiar de criação da população inicial (que utiliza um algoritmo construtivo para tentar selecionar os melhores indivíduos) e no método de reparo de indivíduos inválidos (que tenta manter os indivíduos dentro do espaço de busca factível e ao mesmo tempo tenta selecionar os indivíduos mais aptos); o segundo, é que não é utilizado o operador de cruzamento, comum em um AG canônico; o terceiro é que a solução é representada por uma população e não por um indivíduo. AGs que utilizam seleção-mutação podem ser vistos nos trabalhos (GOMES; GUALDA, 2011) e (DOMBRY, 2007).

Além disso, o AGSOJ/D adiciona duas funções diferentes de aptidão: aptidão de genes e a aptidão da população. Ambos são definidos usando um valor mínimo e ambos são usados como um critério de parada. O Algoritmo 3 mostra o pseudo-código do AGSOJ/D seleção-mutação.

A população inicial é criada utilizando um algoritmo construtivo (GOMES; GUALDA, 2011), que mantém os indivíduos dentro da área factível de busca, chamado de reparo nos indivíduos inválidos. A função de aptidão reflete a distribuição dos genes nos indivíduos, ou seja, avalia os melhores genes que serão selecionados pelos indivíduos. Os critérios de parada do algoritmo obedecem uma das seguintes condições:

- Quando o algoritmo tiver uma população com aptidão maior ou igual ao percentual definido pelo usuário e todos os genes da população tiver aptidão maior ou igual ao percentual mínimo também definido pelo usuário, ou seja, o usuário define um intervalo [mín., máx.] de aptidão que os indivíduos devem possuir para o algoritmo convergir ou;

Algoritmo 3: AGSOJ/D seleção-mutação

```
Entrada: AG Parâmetros, Lista de distritos, Lista de oficiais de justiça
1  begin
2      Pop = Criar_População_Inicial(x)
3      Reparar_Individuos_Inválidos(Pop)
4      Avaliar(Pop)
5      while critério de parada não for atingido do
6          Executar_Elitismo(Pop)
7          melhor_Pop = Atualizar_População(Pop)
8          Executar_Mutação(Pop)
9          Reparar_Individuos_Inválidos(Pop)
10         Avaliar(Pop)
11     end
12 end
13 return melhor_Pop
```

- Quando o algoritmo executar n gerações informadas pelo usuário.

A seleção para elitismo é feita de maneira aleatória sobre os melhores indivíduos da população com a finalidade de selecionar probabilisticamente os melhores indivíduos para a próxima geração. A atualização da população compara a população atual com a melhor população. A atualização só ocorre se a população atual for melhor que a população guardada anteriormente. Em seguida, é executado o operador de mutação genética e logo após o algoritmo construtivo de reparo nos indivíduos inválidos. Depois, a população válida é avaliada e o ciclo do algoritmo continua até que o critério de parada seja atendido. Ao final, o algoritmo retorna a melhor população como solução ao PSOJ/D.

4.1.1 Representação do PSOJ/D

A representação cromossômica é fundamental para o AG, sendo a maneira de traduzir a informação do problema em uma solução viável para tratamento pelo computador. Neste contexto, cada cromossomo ou indivíduo é binário. Diferentemente do AG canônico, apresentado na Seção 2.1, onde um indivíduo representa uma solução completa, no AGSOJ/D seleção-mutação o indivíduo representa parte da solução do PSOJ/D e a solução completa é representada pela melhor população. Cada solução tem incorporado em si mesma as seguintes informações:

- Um individuo representa um distrito e uma solução parcial do PSOJ/D;
- Um gene representa um oficial de justiça, que é a menor representação do PSOJ/D;
- O tamanho do cromossomo é o numero de oficiais de justiça que participam da distribuição;

- Um gene igual a 1 indica que o oficial de justiça foi selecionado para aquele distrito e 0 indica que não foi selecionado;
- A população representa a distribuição completa dos oficiais de Justiça disponíveis em todos os distritos e o elitismo da melhor população representa a solução do PSOD/D;
- O tamanho da população é o número de distritos que participam da distribuição;
- Capacidade de genes é a quantidade de genes iguais a 1 que deve possuir cada indivíduo. É uma restrição que o algoritmo deve obedecer porque indivíduos fora da capacidade de genes são soluções inválidas;

A Tabela 3 apresenta a representação do Cenário 1 - Central de Mandado de Imperatriz em AGSOJ/D. Neste cenário, há 6 distritos (6 indivíduos que representam 6 soluções parciais) e 27 oficiais de Justiça (genes). Cada indivíduo possui, nos seus cromossomos, 27 genes, com objetivo de tentar selecionar os melhores genes para cada indivíduo. A estrutura cromossômica é composta por 27 genes vezes 6 indivíduos que representa a solução completa do PSOJ/D para o cenário 1. O tamanho dessa população é 6 de indivíduos. Além disso, essa tabela mostra a capacidade de genes, que é a quantidade de oficiais de Justiça que deve ser selecionado por cada distrito. O indivíduo 1 retrata um distrito com 13 oficiais de justiça sendo selecionados para aquele distrito. Também, pode-se notar que todos os indivíduos da população são válidos, pois o total de genes selecionados (*bits* igual 1) para cada indivíduo correspondem com suas respectivas capacidades de genes.

Tabela 3 – Representação do Cenário 1: Imperatriz

Indivíduo	CG	Soluções válidas	Estrutura cromossômica (27 genes)
Indivíduo 1	13	Combin(27, 13) = 20.058.300	000001111110100001111001010
Indivíduo 2	3	Combin(14, 3) = 364	001000000000000110000000000
Indivíduo 3	3	Combin(11, 3) = 165	000000000000001000000110000
Indivíduo 4	3	Combin(8, 3) = 56	100110000000000000000000000
Indivíduo 5	3	Combin(5, 3) = 10	010000000000000000000000101
Indivíduo 6	2	Combin(2, 2) = 1	000000000001010000000000000
Total:	27	$\approx 6,70 * 10^{15}$	Espaço de Busca: $2^{6*27} = 2^{162}$

Legenda: CG = Capacidade de genes

A Tabela 3 apresentada representa, portanto, uma população com 6 indivíduos e cada indivíduo com 27 genes no cromossomo. O espaço de busca que o algoritmo percorre, para encontrar dentre as soluções válidas as melhores, é $2^{6*27} = 2^{162}$, onde 2 é a representação binária (0 e 1), 6 é o total de indivíduos, 27 é o total de genes por indivíduo e 162 é o tamanho da estrutura cromossômica completa para o cenário 1. O espaço de busca de $2^{162} \approx 1,90 * 10^{25}$ é necessário para que o algoritmo possa considerar todas as soluções válidas do problema que é $\approx 6,70 * 10^{15}$.

A Tabela 4, por sua vez, apresenta a representação do Cenário 2 - Central de Mandado de São Luís em AGSOJ/D. Para esse cenário, há 16 distritos (16 indivíduos que representam 16 soluções parciais) e 84 oficiais de Justiça (genes). Cada indivíduo possui, nos seus cromossomos, 84 genes, com objetivo de tentar selecionar os melhores genes para cada indivíduo. A estrutura cromossômica é composta por 84 genes vezes 16 indivíduos que representa a solução completa do PSOJ/D para o cenário 2. O tamanho dessa população é 16 de indivíduos. Devido ao tamanho de 84 genes, a estrutura cromossômica não será mostrada para o cenário 2.

Tabela 4 – Representação do Cenário 2: São Luís

Indivíduo	CG	Soluções válidas	Estrutura cromossômica (84 genes)
Indivíduo 1	3	Combin(84, 3) = 95284	
Indivíduo 2	5	Combin(81, 5) = 25621596	
Indivíduo 3	5	Combin(76, 5) = 18474840	
Indivíduo 4	7	Combin(71, 7) = 1329890705	
Indivíduo 5	7	Combin(64, 7) = 621216192	
Indivíduo 6	5	Combin(57, 5) = 4187106	
Indivíduo 7	5	Combin(52, 5) = 2598960	
Indivíduo 8	6	Combin(47, 6) = 10737573	
Indivíduo 9	3	Combin(41, 3) = 10660	
Indivíduo 10	10	Combin(38, 10) = 472733756	
Indivíduo 11	8	Combin(28, 8) = 3108105	
Indivíduo 12	4	Combin(20, 4) = 4845	
Indivíduo 13	3	Combin(16, 3) = 560	
Indivíduo 14	6	Combin(13, 6) = 1716	
Indivíduo 15	4	Combin(7, 4) = 35	
Indivíduo 16	3	Combin(3, 3) = 1	
Total:	84	$\approx 1,10 * 10^{89}$	Espaço de Busca: $2^{16*84} = 2^{1344}$

Legenda: CG = Capacidade de genes

Na Tabela 4 pode-se notar que, o espaço de busca que o algoritmo percorre, para encontrar dentre as soluções válidas as melhores, é $2^{16*84} = 2^{1344}$, onde 2 é a representação binária (0 e 1), 16 é o total de indivíduos, 84 é o total de genes por indivíduo e 1344 é o tamanho da estrutura cromossômica completa para o cenário 2. O espaço de busca de $2^{1344} \approx 3,80 * 10^{405}$ é necessário para que o algoritmo possa considerar todas as soluções válidas do problema que é $\approx 1,10 * 10^{89}$.

Portanto, percebe-se que grande parte do espaço de busca dos problemas representados, são de soluções inválidas e graças ao método de reparo, essas soluções inválidas são ignoradas no processo de busca.

Os métodos de criação da população inicial e reparo em indivíduos inválidos utilizam distribuição uniforme e tentam selecionar os melhores genes para cada indivíduo. Nota-se, portanto, por causa da distribuição uniforme, que os genes selecionados de cada indivíduos estão dispersos na estrutura cromossômica.

4.1.2 População inicial

A população inicial tenta selecionar somente os melhores genes para cada indivíduo, em seguida executa o reparo para tornar o indivíduo válido, caso o total de genes selecionado seja menor que capacidade de genes do indivíduo (GOMES; GUALDA, 2011). O Algoritmo 4 mostra o pseudo-código para criar a população inicial do AGSOJ/D.

O algoritmo inicia o laço principal na linha 2 que irá iterar com cada distrito envolvido na distribuição. Em seguida, linha 3, é criado a instância do indivíduo a partir do distrito[x] e na linha 4 é criado o cromossomo do indivíduo com todos os genes com valores iguais a zero. A partir da linha 6 começa outro laço secundário que irá tentar selecionar os melhores genes para cada indivíduo. Observa-se que há uma tentativa de selecionar com probabilidade uniforme (50%) somente os genes com aptidão ótima e que ainda não foram selecionados por outro indivíduos (linhas 7 a 9). Essa estratégia tende a gerar indivíduos com genes abaixo de sua capacidade de genes, no entanto isso é contornado com o método de reparo nesses indivíduos inválidos (linhas 14 a 16). Depois de pronto, ou seja, depois de válido, o indivíduo é adicionado à população inicial (linha 17). O método selecionaGene(y) (linha 10) muda o valor do gene y para '1' indicando que esse gene foi selecionado para o indivíduo.

Algoritmo 4: AGSOJ/D: Criação da população inicial

```
   Entrada: Lista de distritos, Lista de oficiais justiça
 1 begin
 2     foreach (x = 0; x < distritos.tamanho(); x++) do
 3         individuo = individuo.novo(distrito[x])
 4         individuo.criaCromossomo("0")
 5         total_genes = 0
 6         foreach (y = 0; y < individuo.cromossomo.tamanho()) do
 7             if individuo.gene_nao_selecionado(y) e
 8             Aleatório(1) >= 0.5 e
 9             individuo.avaliaGene(y) = 100 then
10                 individuo.selecionaGene(y)
11                 total_genes = total_genes + 1
12             end
13         end
14         if total_genes < individuo.capacidade() then
15             individuo.repara()
16         end
17         populacao.adiciona(individuo)
18     end
19 end
20 return populacao
```

Esse algoritmo na verdade é um método do AGSOJ/D que tem a finalidade de criar a população inicial com n indivíduos limitados à quantidade de distritos que estão

participando da distribuição.

4.1.3 Função de aptidão

A função de aptidão deve representar a qualidade de cada indivíduo em relação ao problema que está sendo resolvido, e é a única conexão forte entre o AG e o problema (CORTES et al., 2013), (LINDEN, 2012).

A aptidão do gene é usada para calcular a aptidão do indivíduo e da população. A aptidão do gene é calculada em percentual usando a seleção histórica dos distritos recentes, isto é, considerando a informação histórica de que oficial de Justiça foi selecionado para quais distritos em ordem cronológica de acordo com equação 4.1

$$ag = \begin{cases} 100\% & \text{, se } p = 0 \\ 0\% & \text{, se } p = 1 \\ p * \frac{100}{n} & \text{, senão} \end{cases} \tag{4.1}$$

Onde, ag é a aptidão do gene, p é a posição do distrito, na lista de histórico dos últimos n distritos, n é o numero de distritos participantes na distribuição. Logo, n é o tamanho da população. Se $p = 0$ então $ag = 100\%$ e indica que o oficial de Justiça não foi selecionado pelo distrito nas últimas n distribuições; se $p = 1$ então $ag = 0\%$ e indica que o oficial de justiça está sendo selecionado para um distrito ao qual ele foi selecionado na última distribuição, ou seja, o oficial está repetindo o mesmo distrito em sequência violando a principal restrição do PSOJ/D; caso contrário, a aptidão é calculada proporcional a p em relação a n. No final, ag é um vetor contendo todas as avaliações de genes. Pode-se concluir que quanto maior o valor de p ou $p = 0$, melhores serão os resultados obtidos.

O indivíduo é avaliado pela menor aptidão de seus genes. Se qualquer indivíduo tem um gene com aptidão zero, então zero será a aptidão desse indivíduo.

Finalmente, a população é calculada também em percentual pela equação 4.2, em que ap é a aptidão da população, q é a quantidade de genes com aptidão ótima, isto é, com aptidão igual a 100%, $ag0$ é a quantidade de genes com aptidão péssima, ou seja, aptidão igual a zero, e tc é o tamanho do cromossomo. Logo, a aptidão da população é o percentual de $q - ag0$ genes, que tiveram aptidão ótima, menos aptidão péssima, em relação ao tamanho do cromossomo tc. A variável $ag0$ é necessária para aplicar a penalidade descrita na Seção 3.4.

$$ap = (q - ag0) * \frac{100}{tc} \tag{4.2}$$

A Tabela 5 mostra o histórico das últimas 6 distribuições, de 5 a 10, comparando com a distribuição 11 (atual). O resultado apresentado serve para explicar a equação 4.1,

que é utilizada para calcular a aptidão de cada gene.

Tabela 5 – Resultado panorâmico das distribuições de 5 a 10.

	Últimas 6 distribuições								
Índice	p6	p5	p4	p3	p2	p1			
Gene	5	6	7	8	9	10	Equação 4.1	11 (Atual)	% Aptidão
1	1	2	6	1	6	3	p = 0	5	100,00%
2	5	4	3	1	5	6	p = 0	2	100,00%
3	6	4	1	3	6	2	p = 0	5	100,00%
4	**3**	2	1	5	6	4	$6 * \frac{100}{6}$	3	100,00%
5	5	2	1	**6**	4	3	$3 * \frac{100}{6}$	6	50,00%
6	1	6	1	2	4	**5**	p = 1	5	0,00%
7	6	4	3	6	4	5	p = 0	1	100,00%
8	5	3	**6**	3	2	4	$4 * \frac{100}{6}$	6	66,66%
9	1	5	2	3	6	4	$2 * \frac{100}{6}$	6	33,33%
10	2	**6**	5	4	3	1	$5 * \frac{100}{6}$	6	83,33%

Ao observar a Tabela 5 acima é possível conferir a aptidão dos genes 1 ao 10. Na distribuição 11 (atual), o gene 1 foi selecionado para o indivíduo 5 com a aptidão 100%, pois tal gene ainda não havia sido selecionado para esse indivíduo nas distribuições de 5 a 10, logo $p = 0$.

O gene 4 foi selecionado para o indivíduo 3 com aptidão 100% calculada da seguinte maneira: $6 * \frac{100}{6} = 100\%$, onde 6 é a posição p que encontra-se o indivíduo 6 contando da direita para esquerda nas posições $p1$ a $p6$ que correspondem as distribuições de 10 a 5, respectivamente.

O gene 5 foi selecionado para o indivíduo 6 com aptidão 50%. Essa aptidão é calculada da seguinte forma: aplica-se a equação $3 * \frac{100}{6} = 50\%$, onde 3 é a posição p que encontra-se o indivíduo 6 contando da direita para esquerda nas posições $p1$ a $p6$.

4.1.4 Semi-elitismo

O conceito de semi-elistimo é específico para este trabalho, uma vez que, os indivíduos representam soluções parciais do PSOJ/D. No AGSOJ/D seleção-mutação após a avaliação da população são selecionados aleatoriamente um dos melhores indivíduos para o cenário 1 e dois indivíduos dos melhores para cenário 2 que serão mantidos para próxima geração.

4.1.5 Atualização da melhor população

Atualização da melhor população ocorre com o elitismo aplicado à população que consiste em guardar a melhor solução dentre todas as gerações criadas. O processo de elistimo da melhor população no AGSOJ/D, além de comparar a aptidão da população

é comparado a média de aptidão dos genes e também a menor aptidão dos genes para determinar se a geração corrente é melhor que a última geração guardada. A Tabela 6 apresenta a evolução da melhor população ao longo de 30.000 gerações executadas sobre a instância do cenário 2 pelo AGSOJ/D seleção-mutação e são mostradas somente as gerações em que houve uma melhora na aptidão da população, na média de aptidão dos genes ou na menor aptidão dos genes.

Tabela 6 – Evolução da melhor população

Geração	Aptidão da população	Média de aptidão dos genes	Menor aptidão dos genes	Tempo (ms)
1	0,00	10.75	0,00	0,140
2	55,00	83.20	33,00	0,738
3	60,00	79.05	33,00	0,867
62	60,00	84.05	33,00	5,829
133	60,00	86.55	50,00	11,000
621	60,00	87.40	50,00	57,000
1.924	60,00	88.15	50,00	213,000
1.925	65,00	87.35	50,00	213,000
1.961	65,00	89.00	50,00	218,000
2.174	70,00	89.90	50,00	245,000
18.775	70,00	90.70	50,00	5.542,000

Pode-se observar na Tabela 6 a evolução da melhor população em termos de aptidão da população, média de aptidão dos genes e menor aptidão dos genes. A partir da geração 3ª quando a aptidão da população manteve-se em 60%, nota-se que a atualização da melhor população deu-se pela evolução da média de aptidão dos genes até a geração 1.924. Embora o algoritmo tenha executados 30.000 gerações, mas a melhor solução e mantida na geração 18.775 e retornada pelo algoritmo.

4.1.6 Mutação genética

O operador de mutação é fundamental para um AG, pois é ele que garante a continuidade da existência de diversidade genética na população, enquanto que o operador de cruzamento contribui fortemente para a igualdade entre os indivíduos. O valor da probabilidade que decide se o operador de mutação será ou não aplicado é um dos parâmetros do AG que apenas a experiência pode determinar (LINDEN, 2012).

A mutação genética é aplicada a cada indivíduo da população com base em uma taxa de mutação, exceto sobre aqueles que foram selecionados pelo elitismo na geração anterior (BOLTON et al., 2016), (HASSANAT et al., 2016). Após testes exaustivos para determinar o melhor percentual de mutação, observou-se que o percentual de 50% explora a diversidade de genes por indivíduos. A Tabela 7 ilustra a operação de mutação utilizada no AGSOJ/D.

Tabela 7 – Mutação Genética do AGSOJ/D

Indivíduo	Capacidade de genes	Cromossomo
Indivíduo válido	3	00000100001100000000
Indivíduo após mutação	3	00010111111010011000
Indivíduo reparado	3	00000100101000000000

Observa-se na linha 3 da Tabela 7 que após realizar a operação de mutação o indivíduo passa a ter mais genes selecionados (10 *bits* '1') que sua capacidade de genes igual a 3. Logo torna-se necessário realizar um reparo no indivíduo com a finalidade de mantê-lo dentro da região factível de busca, como pode ser observado na linha 4 da mesma tabela.

4.1.7 Reparo em indivíduos inválidos

Após criar a população inicial e executar o operador de mutação binária é necessário fazer um reparo na estrutura genética do indivíduo afim de corrigir indivíduos com genes inválidos. Isto se deve à natureza estocástica desta metodologia, que torna os resultados obtidos diferentes a cada geração do algoritmo. O artigo de GOMES e GUALDA executa uma busca local nos melhores indivíduos com objetivo de melhorar a aptidão dos mesmos (GOMES; GUALDA, 2011), (WANG; ACHARYA; KAM, 2016). A função de reparo, além de tentar melhorar a aptidão dos indivíduos tenta corrigir os genes de cada indivíduo deixando-os como prováveis soluções para o problema.

O Algoritmo 5 inicia-se recebendo como parâmetro um indivíduo inválido após o mesmo ser criado pelo método de criação da população inicial. No primeiro laço, o algoritmo repara a estrutura cromossômica removendo genes além da capacidade, após mutação genética, quando $total\_genes = 0$, linha 2. Além disso, nesse laço o algoritmo confere quantos genes selecionados existem na estrutura cromossômica do indivíduo que servirá para o controle do segundo laço iniciado na linha 15.

No segundo laço, o processo de seleção dos melhores genes é iterativo e são executadas 100 iterações de reparação tentando selecionar os melhores genes para o indivíduo e se ao final de 100 iterações o indivíduo ainda apresentar aptidão ruim, então ele é adicionado à população mesmo com aptidão zero. Logo, o laço termina quando o total de genes selecionados para indivíduo é completado ou quando são executadas 100 iterações.

De acordo com o Algoritmo 5 apresentado, nota-se que o reparo é feito de duas maneiras diferentes: quando o individuo possuir mais genes selecionados (alelo igual a 1) que sua capacidade de genes (a função de reparo irá eliminar genes da sua estrutura), primeiro laço, linha 3 e caso contrário quando indivíduo possuir menos genes com alelo 1 que sua capacidade de genes (a função de reparo irá tentar selecionar os melhores genes para completar a sua estrutura), segundo laço, linha 15. O reparo é concluído quando o

Algoritmo 5: AGSOJ/D: Algoritmo de reparo em indivíduos inválidos

```
   Entrada: Indivíduo inválido
1  begin
2      if total_genes = 0 then
3          foreach (y = 0; y < individuo.cromossomo.tamanho(); y++) do
4              if individuo.gene_selecionado(y) then
5                  if total_genes < individuo.capacidade() then
6                      total_genes = total_genes + 1
7                  else
8                      individuo.removeGene(y)
9                  end
10             end
11         end
12     end
13     vezes = 1
14     i = 1
15     while vezes <= 100 e total_genes < individuo.capacidade() do
16         if individuo.gene_nao_selecionado(y) e
17         Aleatório(1) >= 0.5 then
18             aptidao = individuo.avaliaGene(y)
19             if (vezes >= 100 ou aptidao > 100 - vezes) then
20                 individuo.selecionaGene(y)
21                 total_genes = total_genes + 1
22             end
23             i = i + 1
24             if i > individuo.cromossomo.tamanho() then
25                 i = 1
26                 vezes = vezes + 1
27             end
28         end
29     end
30 end
31 return individuo
```

indivíduo torna-se válido, ou seja, a quantidade de genes com alelo 1 (selecionado) torna-se igual a capacidade de genes do indivíduo. A Tabela 8 ilustra um reparo realizado em um indivíduo inválido.

Tabela 8 – Reparo em indivíduo com genes inválidos

Indivíduo	Capacidade de genes	Cromossomo
Indivíduo inválido	5	00000000110000000000
Indivíduo reparado	5	10000100110000000010

A capacidade de genes do indivíduo é de 5 genes e após a operação de mutação ou criação da população inicial o indivíduo tornou-se inválido, pois passou a ter apenas 2 genes

selecionados (2 *bits* '1'). Então a função de reparo tenta acrescentar 3 genes com melhor aptidão a sua estrutura. Se não conseguir os melhores genes, então seleciona qualquer gene.

Após a pesquisa bibliográfica sobre a meta-heurística de AG foi possível fazer a representação binária do PSOJ/D e definir os principais componentes do AGSOJ/D, como: criação da população inicial, função de aptidão, elitismo, atualização da população, mutação genética e o método de reparo que possui o algoritmo mais relevante para essa solução. A seguir será apresentado uma solução ao PSOJ/D acrescentando o operador de cruzamento ao AGSOJ/D.

4.2 AGSOJ/D-X com cruzamento

Historicamente, o operador de cruzamento tem recebido uma percentagem de escolha muito alta, variando de 60% a 95% com taxas de mutação bastante baixas, variando de 0,5% a 1%, reforçando a importância do operador de cruzamento na geração de soluções (LINDEN, 2012). A seleção dos pais para cruzamento foi feita utilizando a seleção por roleta.

O AGSOJ/D teve uma modificação na sua estrutura para incluir o operador de cruzamento e para diferenciá-lo da solução original proposta foi chamado de AGSOJ/D-X. O cromossomo do AGSOJ/D-X, diferentemente do AGSOJ/D seleção-mutação que retrata uma solução parcial para o PSOJ/D, é formado com 162 genes para o cenário 1 que equivale a 6 distritos multiplicado por 27 oficiais de justiça, e 1344 genes para o cenário 2 equivalente a 16 distritos vezes 84 oficiais de justiça e cada cromossomo representa uma solução completa para o PSOJ/D. O indivíduo com melhor qualidade de aptidão é retornado como solução ao PSOJ/D.

O tamanho da população foi definido com 6 indivíduos para o cenário 1 e 16 indivíduos para o cenário 2. A taxa de cruzamento igual a 80% e taxa de mutação máxima recomendada de 1%. Os demais parâmetros de entradas foram mantidos para uma comparação justa entre AGSOJ/D seleção-mutação e o AGSOJ/D-X com cruzamento.

Experimentos nos cenários 1 e 2 foram realizados no AGSOJ/D-X e confrontados com o AGSOJ/D seleção-mutação e SA, cujo principal objetivo foi provar que para o PSOJ/D o operador de cruzamento é inapropriado, pois tornou o algoritmo mais lento e ineficaz em obter boas soluções, de acordo com diversos testes previamente realizados. A Tabela 9 ilustra a operação de cruzamento utilizada no AGSOJ/D-X.

No exemplo de cruzamento demonstrado na Tabela 9, meramente ilustrativo por questões de entendimento, pode-se observar que trata-se de um cruzamento de 1 ponto, onde os genes antes do corte do Pai 1 concatenado com os genes após o corte do Pai 2 formam o Filho 1 e os genes após o corte do Pai 1 concatenado com os genes antes do

Tabela 9 – Exemplo de cruzamento do AGSOJ/D-X

Indivíduo	Cromossomo
Pai 1	00000100001|100000000
Pai 2	00010000000|000011000
Filho 1	00000100001|000011000
Filho 2	100000000|00010000000

corte do Pai 2 formam o Filho 2.

A seguir será apresentado como foi representada a solução ao PSOJ/D utilizando a meta-heurística de SA.

4.3 Têmpera Simulada

Uma solução em SA do PSOJ/D é representada por uma lista de distritos e cada distrito contém uma lista de oficiais de justiça selecionados de acordo com suas respectivas capacidades. Dessa forma, cada solução criada ou modificada ao longo da execução do algoritmo SA irá sempre representar uma solução válida para o PSOJ/D.

A solução inicial é criada aleatoriamente com base na lista de distritos e oficiais de justiça recebidas como parâmetros pelo SA. A solução é avaliação pela aptidão de cada oficial de justiça proporcionalmente em relação à quantidade de oficiais de justiça que participam da distribuição. A aptidão do oficial de justiça é calculada em percentual de acordo com a equação 4.3.

$$aoj = \begin{cases} 100\% & \text{, se } p = 0 \\ 0\% & \text{, se } p = 1 \\ p * \frac{100}{n} & \text{, senão} \end{cases} \tag{4.3}$$

Onde, aoj é a aptidão do oficial de justiça; p é a posição do distrito, na lista de histórico dos últimos n distritos, n é o numero de distritos participantes na distribuição. Se $p = 0$ então $aoj = 100\%$ e indica que o oficial de Justiça não foi selecionado pelo distrito nas últimas n distribuições; se $p = 1$ então $aoj = 0\%$ e indica que o oficial de justiça está sendo selecionado para um distrito ao qual ele foi selecionado na última distribuição, ou seja, o oficial está repetindo o mesmo distrito em sequência violando a principal restrição do PSOJ/D; caso contrário, a aptidão é calculada proporcional a p em relação a n.

A aptidão da solução é calculada também em percentual pela equação 4.4, em que as é a aptidão da solução, q é a quantidade de oficiais de justiça com aptidão ótima, isto é, com aptidão igual a 100%, $a0$ é a quantidade de oficiais de justiça com aptidão péssima, ou seja, aptidão igual a zero, e qoj é a quantidade de oficiais de justiça. Logo, a aptidão da solução é o percentual de q oficiais de justiça, que tiveram aptidão ótima, menos

aptidão péssima, em relação à quantidade de oficiais de justiça qoj que participantes da distribuição. A variável $a0$ é necessária para aplicar a penalidade descrita na Seção 3.4.

$$as = (q - a0) * \frac{100}{qoj} \tag{4.4}$$

Com exceção dessas particularidades apresentadas, o algoritmo SA implementado para solucionar o PSOJ/D é o mesmo descrito na Seção 2.2 sobre Simulated Annealing.

A seguir serão apresentados os experimentos realizados nas três abordagens apresentadas, bem como as discussões sobre os resultados obtidos nos dois cenários.

4.4 Considerações finais

Este capítulo apresentou as três soluções para o PSOJ/D da seguinte forma: a Seção 4.1 mostrou detalhes do AGSOJ/D seleção-mutação, como o problema foi modelado e adaptado para solucionar o PSOJ/D; a Seção 4.2 apresentou a representação do PSOJ/D no AGSOJ/D-X com cruzamento e finalmente a Seção 4.3 descreveu como foi modelado o PSOJ/D em *Simulated Annealing*.

5 Experimentos Computacionais

Os experimentos foram executados em um notebook contendo um processador de 4ª geração da Intel® Core™ i7 - 4500U (1.8 GHz até 3.0 GHz, cache de 4 M), 08 GB de Memória RAM e Disco Rígido 1TB, SATA (5400 RPM), executando o Sistema Operacional Windows 8.1, 64bits.

Como mencionado anteriormente, os experimentos foram realizados em dois cenários diferentes, usando os parâmetros mostrados na tabela 10 para algoritmos de AGSOJ/D: seleção-mutação, AGSOJ-X: com cruzamento e SA. Foram Realizadas 10 distribuições dos oficiais de justiça para ambos os cenários. Cada distribuição com 30 execuções dos Algoritmos AGSOJ/D seleção-mutação, AGSOJ/D-X com cruzamento e SA. O teorema central do limite, afirma que qualquer amostra a partir de 30 instâncias tende a apresentar uma distribuição normal (MONTGOMERY; RUNGER, 2003).

A cada execução o AGSOJ/D seleção-mutação e o AGSOJ/D-X com cruzamento foram submetidos a 10.000 gerações para o cenário 1 e 30.000 gerações para o cenário 2; o SA também foi submitido a 10.000 avaliações para o cenário 1 e 30.000 avaliações para o cenário 2.

A melhor solução de cada execução do AGSOJ/D seleção-mutação, do AGSOJ/D-X e do SA foram armazenadas em base de dados. Depois de realizados todos os experimentos nos dois cenários nas três abordagens, os resultados obtidos foram planificados para calcular as médias, o desvio padrão e realizar o Teste-T Student. Também foram obtidos resultados de distribuições manuais, referente ao cenário 1 que foram comparados com os resultados aptidões obtidos do AGSOJ/D seleção-mutação.

Tabela 10 – Parâmetro de entradas AGSOJ/D, AGSOJ-X e SA por cenário

AGSOJ/D: seleção-mutação		
Parâmetros de entrada	Cenário 1	Cenário 2
Taxa de mutação	50%	50%
Número de gerações	10.000	30.000
AGSOJ/D-X: com cruzamento		
Parâmetros de entrada	Cenário 1	Cenário 2
Taxa de cruzamento	80%	80%
Taxa de mutação	1%	1%
Número de gerações	10.000	30.000
SA		
Parâmetros de entrada	Cenário 1	Cenário 2
Número de avaliações	10.000	30.000

Concernente à comparação contra SA, uma solução é formada por um cromossomo

com 162 genes para o primeiro cenário que equivale a 6 distritos multiplicado por 27 oficiais de justiça, e 1344 genes para o segundo cenário equivalente a 16 distritos vezes 84 oficiais de justiça. Esta modificação é necessária por duas razões. A primeira é para fazer a comparação justa em termos de iterações. A segunda é para ter uma solução completa, que envolvam todos os oficiais de justiça e todos os distritos.

A fim de mostrar que o AGSOJ/D é melhor que SA, foi realizado um Teste-T bi-caudal com nível de significância $\alpha = 0,95$, considerando H_0 como sendo "todas as médias são semelhantes"; portanto, qualquer valor fora do intervalo $[-2,0452; 2,0452]$ é considerado como uma rejeição de H_0, ou seja há uma diferença significativa entre as médias.

5.1 Cenário 1: 6 indivíduos x 27 genes

A Tabela 3 retrata o problema da Central de Mandados de imperatriz com 6 distritos e 27 oficiais de justiça. Os dados constantes dessa tabela foram extraídos da base dados do sistema de acompanhamento processual Themis.

A Figura 12 mostra as aptidões de 10 distribuições realizadas manualmente, referente ao período de 01/08/2013 a 28/02/2016, sendo uma distribuição a cada três meses. Os valores de aptidões foram extraídos da base de dados do sistema Themis e representa valores de distribuições reais existentes. Esses valores estão sendo comparados com a média aptidão de 10 distribuições realizadas pelo AGSOJ/D seleção-mutação, considerando como histórico as distribuições realizadas manualmente, ou seja, as distribuições não foram realizadas em base de dados zeradas.

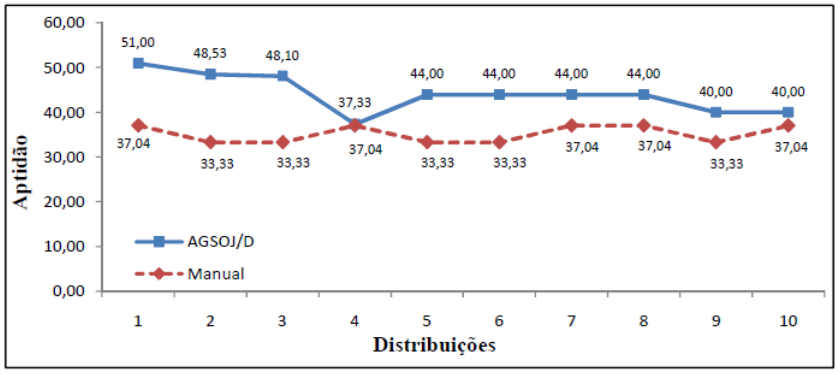

Figura 12 – Cenário 1: AGSOJ/D x Manual - Aptidão

Os resultados apresentados na Figura 12 mostram que o AGSOJ/D seleção-mutação, além de ser incomparavelmente mais rápido que o processo manual, também mostrou-se mais eficiente em obter soluções melhores que o processo manual.

A Figura 13 que mostra a curva de comparação da média aptidão das 10 distribuições

realizadas para o cenário 1 evidencia e sustenta a tese de que o AGSOJ/D seleção-mutação fora bem sucedido após os experimentos realizados quanto ao aspecto de aptidão comparado com o AGSOJ/D-X e SA.

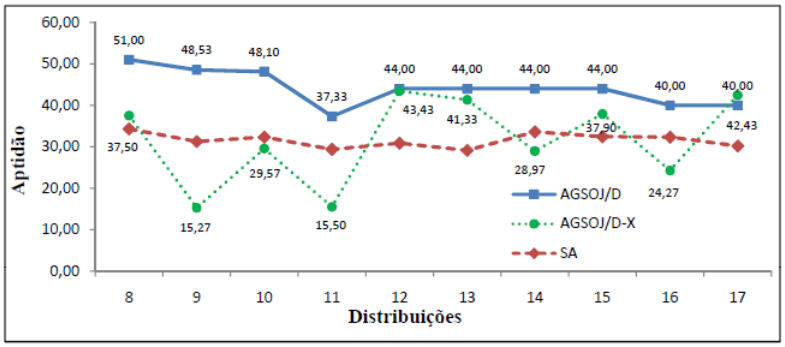

Figura 13 – Cenário 1: Média de Aptidão

Os mesmos resultados do AGSOJ/D seleção-mutação apresentados na Figura 13 são os que foram comparados com o processo manual.

O desvio padrão das 30 aptidões em 10 distribuições realizadas, mostrado na Figura 14, comprova o quanto o AGSOJ/D seleção-mutação manteve-se estável comparado com ao AGSOJ/D-X e ao SA.

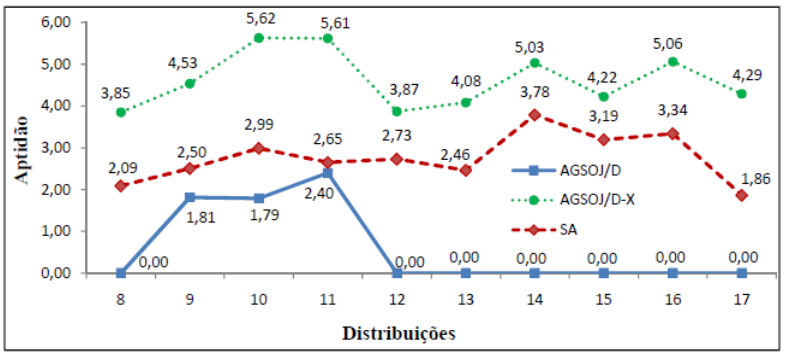

Figura 14 – Cenário 1: Desvio Padrão

O AGSOJ/D seleção-mutação teve uma pequena instabilidade tolerável somente nas distribuições 9, 10 e 11, nas demais distribuições mostrou-se 100% estável.

A Tabela 11 mostra um Teste-T bi-caudal com $\alpha = 0,95$ e $H_0 = [-2,0452; 2,0452]$, comparando AGSOJ/D seleção-mutação contra SA em relação às médias de aptidões obtidas nos experimentos do cenário 1.

O $p - valor$ fornece a probabilidade dos resultados serem efeitos do fator sorte. Portanto, pode-se concluir que, para qualquer nível de significância maior que $0,000001464$, tem-se evidências para rejeitar a hipótese nula em todas as distribuições com nível de segurança de 95% e os valores do Teste-T indicam que há uma diferença significativa entre as duas médias comparadas no cenário 2, de acordo com a Tabela 11.

Tabela 11 – Cenário 1: Teste-T - AGSOJ/D X SA

	Média de Aptidão		Desvio Padrão		
Distribuições	AGSOJ/D	SA	AGSOJ/D	SA	Teste-T
8	51,00	34,30	0,00	**2,09**	43,83
9	48,53	31,27	1,81	**2,50**	27,54
10	48,10	32,33	1,79	**2,99**	21,70
11	37,33	29,40	2,40	**2,65**	14,14
12	44,00	30,87	0,00	**2,73**	26,39
13	44,00	29,13	0,00	**2,46**	33,10
14	44,00	33,60	0,00	**3,78**	15,05
15	44,00	32,43	0,00	**3,19**	19,85
16	40,00	32,33	0,00	**3,34**	12,59
17	40,00	30,20	0,00	**1,86**	28,79
					$p - valor = 0,000001464$

5.2 Cenário 2: 16 indivíduos x 84 genes

O cenário 2 retrata o problema da distribuição de oficiais de justiça na central de mandados de São Luís com 16 distritos e 84 oficiais de justiça. Os dados constantes na Tabela 4 foram extraídos da base dados do sistema de acompanhamento processual Themis.

A Figura 15 mostra as médias de aptidões das três abordagens experimentadas e nota-se ao longo das 10 distribuições a superioridade do AGSOJ/D seleção-mutação em relação ao AGSOJ/D-X e ao SA em obter boas soluções.

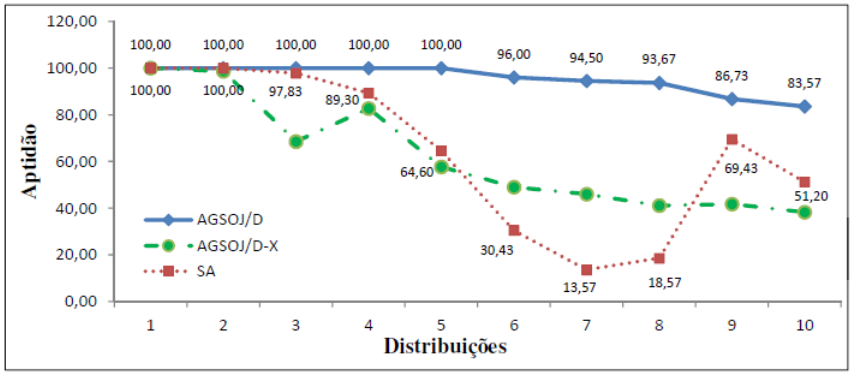

Figura 15 – Cenário 2: Média de Aptidão

Para o cenário 2, o AGSOJ/D obteve ótimo global até a 5ª distribuição, enquanto o SA obteve ótimo global somente até a 2ª distribuição. Um fato interessante a destacar é que para o cenaário 2 os experimentos nas três abordagens foram realizadas em base de dados zeradas, ou seja, todos os experimentos iniciaram da primeira distribuição.

De acordo com a Figura 16, o AGSOJ/D seleção-mutação manteve-se estável em todas as distribuições realizadas.

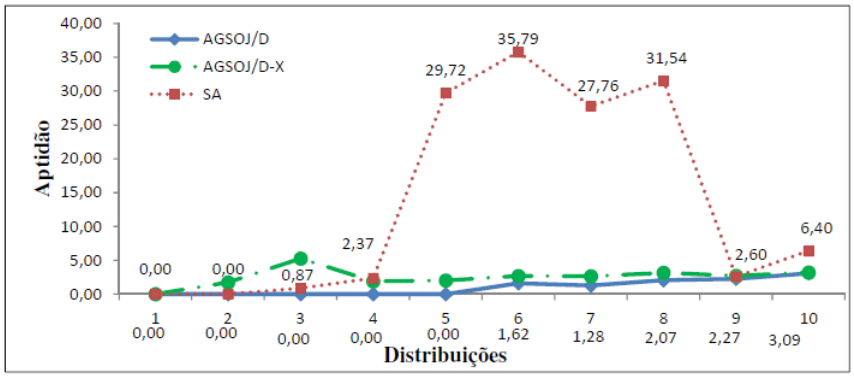

Figura 16 – Cenário 2: Desvio Padrão

Pode-se observar um nível de confiança aceitável em obter soluções ótimas ou próximas do ótimo. Por outro lado o SA, assim como no cenário 1, mostrou-se instável com pico de desvio padrão de 35% na 6ª distribuição e o AGSOJ/D-X embora tenha mantido-se estável em quase todas as distribuições não foi capaz de obter resultados próximos ou melhores que o AGSOJ/D seleção-mutação. Conclui-se portanto a estabilidade e eficiência do AGSOJ/D seleção-mutação no cenário real da Central de Mandados de São Luís.

A Tabela 12 apresenta um Teste-T bi-caudal com $\alpha = 0,95$ e $H_0 = [-2,0452; 2,0452]$, que evidencia uma comparação para o cenário 2 entre o AGSOJ/D seleção-mutação e SA, no qual pode-se ver as diferenças começam a ser significativas a partir da 3ª distribuição e tais diferenças são 95% confiáveis.

Tabela 12 – Cenário 2: Teste-T - AGSOJ/D X SA

	Média de Aptidão		Desvio Padrão		
Distribuições	AGSOJ/D	SA	AGSOJ/D	SA	Teste-T
1	100,00	100,00	0,00	0,00	0,00
2	100,00	100,00	0,00	0,00	0,00
3	100,00	97,83	0,00	0,87	13,57
4	100,00	89,30	0,00	**2,37**	24,77
5	100,00	64,60	0,00	**29,72**	6,52
6	96,00	30,43	1,62	**35,79**	10,16
7	94,50	13,57	1,28	**27,76**	15,98
8	93,67	18,57	2,07	**31,54**	12,83
9	86,73	69,43	2,27	**2,60**	35,02
10	83,57	51,20	3,09	**6,40**	26,03
					$p - valor = 0,010907$

Um fato interessante é quanto mais o problema se torna mais difícil de resolver, o AGSOJ/D seleção-mutação permanece mais estável, apresentando diferenças mais significativas entre a quinta e oitava distribuições.

Pode-se concluir que, para qualquer nível de significância maior que $0,010907$, tem-se evidências para rejeitar a hipótese nula a partir da 3ª distribuição com nível de

segurança de 95% e os valores do Teste-T indicam que há uma diferença significativa entre as duas médias comparadas no cenário 2, de acordo com a Tabela 12.

Para maior clareza em relação às médias de aptidões apresentadas no gráficos para os cenários 1 e 2 as Tabelas 13 e 13 apresentam as médias de aptidões das três abordagens experimentadas: AGSOJ/D seleção-mutação, AGSOJ/D-X e SA.

Tabela 13 – Cenário 1: Resultados das médias de aptidão

Distribuições	AGSOJ/D	AGSOJ/D-X	SA
8	100,00	100,00	100,00
9	100,00	100,00	100,00
10	100,00	56,83	96,17
11	100,00	57,50	73,50
12	95,00	54,50	61,67
13	76,83	52,33	52,17
14	85,00	48,83	63,50
15	80,00	43,67	49,17
16	79,83	67,67	63,33
17	79,50	76,17	51,50
Média	89,62	65,75	71,10

Tabela 14 – Cenário 2: Resultados das médias de aptidão

Distribuições	AGSOJ/D	AGSOJ/D-X	SA
1	100,00	100,00	100,00
2	100,00	98,67	100,00
3	100,00	68,50	97,83
4	100,00	82,70	89,30
5	100,00	57,70	64,60
6	96,00	48,97	30,43
7	94,50	45,97	13,57
8	93,67	41,10	18,57
9	86,73	41,73	69,43
10	83,57	38,23	51,20
Média	95,45	62,36	63,49

Como pode ser visto as soluções obtidas pelo AGSOJ/D seleção-mutação são melhores que as outras duas abordagens comparadas nos dois cenários.

5.3 Tempos de execução

É notória a diferença do *Simulated Annealing* em termo de performance comparado com o AGSOJ/D seleção-mutação, conforme os resultados apresentados em segundos nas Tabelas 15 e 16. Ao observar o cenário 2 na 7ª distribuição, o AGSOJ/D seleção-mutação demorou mais de um minuto, 63,410s, contra apenas 5,683s do SA que são as médias de

tempo de execuções, referente a 30 execuções de cada algoritmo. Contudo o SA obteve sua pior média de aptidão nessa distribuição, 13,57%, contra 94,50% do AGSOJ/D seleção-mutação. O que justifica e compensa o tempo a mais gasto pelo AGSOJ/D seleção-mutação. Por outro lado o AGSOJ/D-X comparado ao AGSOJ/D seleção-mutação, manteve-se mais rápido no cenário 1, porém no cenário 2 a perda de performance e os resultados obtidos são bastante significantes.

Tabela 15 – Cenário 1: Resultados dos tempos de execução em segundos

Distribuições	AGSOJ/D	AGSOJ/D-X	SA
8	2,007	2,357	0,225
9	2,118	2,682	0,222
10	2,095	2,455	0,231
11	2,160	2,133	0,246
12	2,161	1,383	0,229
13	2,396	1,009	0,230
14	2,274	1,051	0,262
15	2,309	1,116	0,245
16	2,342	1,060	0,221
17	2,279	0,939	0,246
Média	1,269	1,046	0,112

Tabela 16 – Cenário 2: Resultados dos tempos de execução em segundos

Distribuições	AGSOJ/D	AGSOJ/D-X	SA
1	0,000	0,033	0,001
2	0,001	3,181	0,042
3	0,040	116,311	4,847
4	0,765	86,640	4,962
5	0,935	97,666	5,280
6	40,520	92,117	5,734
7	63,410	94,394	5,683
8	41,434	102,854	5,644
9	45,904	104,801	4,903
10	45,994	109,765	5,144
Média	23,900	80,776	4,224

5.4 Considerações finais

Os experimentos e discussões apresentadas consolidam a eficiência do AGSOJ/D seleção-mutação em obter sempre as melhores soluções para o PSOJ/D de acordo com as médias de aptidões demonstradas e comparadas em relação ao AGSOJ/D-X e *Simulated Annealing*. A desvantagem do AGSOJ/D seleção-mutação em relação ao SA foi o tempo de resposta, pois o SA tem uma implementação mais simples que o AGSOJ/D seleção-mutação. Contudo essa desvantagem é aceita quando confrontada com a qualidade dos

resultados obtidos. Em comparação com AGSOJ/D-X, o AGSOJ/D seleção-mutação só apresentou desvantagem no cenário 1 em relação à performance. A seguir será apresentado as conclusões e recomendação para trabalhos futuros.

6 Conclusões

Por meio dos estudos de trabalhos que utilizam as meta-heurísticas de Algoritmo Genético - AG e *Simulated Annealing - SA*, de ferramentas de desenvolvimento em JAVA e de outras referências bibliográficas foi possível desenvolver o projeto de *software* AGSOJ/D para Web com uma nova meta-heurística chamada AGSOJ/D seleção-mutação. Além disso, foram desenvolvidas outras duas soluções, o AGSOJ/D-X com cruzamento e o SA, que serviram para comparar com os resultados obtidos pelo AGSOJ/D seleção-mutação. A comparação confirma que os resultados obtidos através do AGSOJ/D seleção-mutação são aceitáveis e confiáveis e resolve o problema de forma satisfatória como pode ser observado nos gráficos de curvas de comparação das médias de aptidões apresentadas.

Os experimentos realizados e os resultados discutidos confirmam a eficiência e robustez do AGSOJ/D seleção-mutação para o PSOJ/D, pois provam que os oficiais de justiça estão sendo distribuídos conforme Provimento 18/2011 da CGJ-MA (PROVIMENTO 18, 2011).

Por causa de sua característica probabilística os AGs não garantem sempre os mesmos resultados a cada execução. Contudo, os gráficos e tabelas apresentadas no Capítulo 5 provam que o AGSOJ/D seleção-mutação, embora sem o operador de cruzamento, manteve-se confiável em obter bons resultados de aptidões para o cenário 1 e melhor ainda para o cenário 2 se comparados com os resultados de aptidões obtidos pelo AGSOJ/D-X e SA. Apesar do SA ter se mostrado mais rápido, as soluções obtidas pelo o AGSOJ/D seleção-mutação foram muito mais eficientes e estáveis.

A solução proposta pode ser facilmente aplicada a outros Tribunais de Justiça que utilizem o conceito de Central de Mandados, mesmo com outro gerenciador de banco de dados diferente do que foi utilizado nesse trabalho. O algoritmo AGSOJ/D seleção-mutação pode ser aplicado a outros problemas semelhantes de otimização, como por exemplo: escala de policiais por ronda. Onde, cada policial irá representar um gene, cada ronda irá representar um indivíduo e a população irá representar uma solução completa para esse problema.

Além disso, os resultados indicam a possibilidade de utilizar o AGSOJ/D em instâncias maiores que contenham mais distritos e mais oficiais de justiça, uma vez que, quanto mais o problema torna-se difícil de resolver, o AGSOJ/D seleção-mutação permanece mais estável, apresentando diferenças mais significativas em relação ao AGSOJ/D-X e ao SA, principalmente no cenário 2, em que o espaço de busca é 2^{1344} com $\approx 1,10 * 10^{89}$ possibilidades de soluções válidas.

Recomenda-se para trabalhos futuros a representação inteira para o AGSOJ/D, ao

invés de binária, permitindo desse modo distribuir um oficial de justiça em mais de um distrito, ou seja, permitir distribuir o mesmo gene em indivíduos diferentes. A Tabela 17 ilustra a representação inteira de uma população com 3 indivíduos e 9 genes (inteiros de 1 a 9). Cada gene representa o código do oficial de justiça, logo permitindo repetição do mesmo em distritos diferentes, pois o cromossomo de cada indivíduo é composto por 5 genes. Esse tipo de situação é possível, porque há casos que determinados oficiais de justiça atendem mais de um distrito por período.

Tabela 17 – Representação inteira para o AGSOJ/D

Indivíduo	Capacidade de genes	Cromossomo								
Indivíduo 1	5			3		9		2	5	1
Indivíduo 2	5	8			2	4	6			7
Indivíduo 3	5	5	3		1			8		6

Nota-se que há repetições de 6 genes (1, 2, 3, 5, 6 e 8) em indivíduos diferentes com objetivo de completar a estrutura cromossômica dos indivíduos.

Referências

BASGALUPP, M. P. *LEGAL-Tree um Algoritmo Genético multi-objetivo para indução de árvores de decisão*. 94f. p. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação ICMC - USP, São Carlos - SP, fev. 2010. Citado 9 vezes nas páginas 17, 23, 24, 25, 26, 27, 28, 29 e 31.

BHAWNA, K.; KUMAR, G.; BHATIA, P. K. Software test case reduction using genetic algorithm: A modified approach. *IJISET - International Journal of Innovative Science, Engineering & Technology*, v. 3, n. 5, p. 6p., May 2016. Citado na página 15.

BOLTON, C. C. et al. A multi-operator genetic algorithm for the generalized minimum spanning tree problem. *Expert Systems With Applications*, Departamento de Ingeniería Informática, Universidad de Santiago de Chile, Av. Ecuador, 3659 Santiago. Chile, n. 50, p. 1–8, 8p., 2016. Citado na página 47.

BÉRARD, J. Genetic algorithms in random environments: Two examples. *Probability Theory and Related Fields*, n. 133, p. 123–140, 2005. Citado na página 15.

CHAVES, A. A. *UMA META-HEURÍSTICA HÍBRIDA COM BUSCA POR AGRUPAMENTOS APLICADA A PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA*. 196f. p. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos - SP, mar. 2009. Citado na página 19.

CORREGEDORIA GERAL DA JUSTIçA – CGJ-MA. *PROVIMENTO 18/2011*. São Luís-MA, 2011. 4 p. Citado 3 vezes nas páginas 14, 34 e 61.

CORTES, O. A. C. et al. Um algoritmo híbrido paralelo cooperativo baseado em de, pso e ag : Uma avaliação em computadores multicore. *Congresso Brasileiro em Inteligência Computacional (CBIC)*, Porto de Galinhas-PE, 2013. Citado 3 vezes nas páginas 15, 22 e 45.

CORTES, O. A. C.; SILVA, J. C. A local search algorithm based on clonal selection. eleventh brazilian symposium on neural networks. p. 6p., 2010. Citado na página 15.

DOMBRY, C. A weighted random walk model, with application to a genetic algorithm. Universite Claude Bernard- Lyon 1, 50, av. Tony-Garnier - 69366 Lyon Cedex, France, p. 21p., 2007. Citado na página 40.

FREITAS, J. *Oficial de Justiça / Central de Mandados*. 2016. [Online; acessado em 18 abr. 2016]. Disponível em: <http://www.oficialdejustica.net.br/central-de-mandados.htm>. Citado 2 vezes nas páginas 34 e 35.

GOMES, P. V.; SARAIVA, J. T. Hybrid genetic algorithm for multi-objective transmission expansion planning. Porto, Portugal, p. 6p., 2016. Citado na página 15.

GOMES, W. d. P.; GUALDA, N. D. F. Modelagem integrada do problema de programação de tripulantes de aeronaves. *Transportes*, São Paulo, v. 19, n. 1, p. 23–32, maio 2011. Citado 5 vezes nas páginas 14, 22, 40, 44 e 48.

HASSANAT, A. et al. Enhancing genetic algorithms using multi-mutations. IT Department, Mu'tah University, Mu'tah – Karak, Jordan, p. 17p., 2016. Citado na página 47.

HEINEN, M. R.; OSÓRIO, F. S. Algoritmos genéticos aplicados ao problema de roteamento de veículos. Universidade do Vale do Rio dos Sinos - UNISINOS: Computação Aplicada - PIPCA, São Leopoldo - RS, p. 7p., 2006. Citado na página 15.

HU, L. et al. A phoenix++ based new genetic algorithm involving mechanism of simulated annealing. *International Journal of Distributed Sensor Networks*, Hubei Co-Innovation Center of Basic Education IT Services, Hubei University of Education, China, v. 2015, n. Article ID 806708, p. 8p., 2015. Citado na página 32.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science, New Series*, v. 220, n. 4598, p. 671–680, 1983. Citado na página 32.

KRAUSE, J.; PARPINELLI, R. S.; LOPES, H. S. Proposta de um algoritmo inspirado em evolução diferencial aplicado ao problema multidimensional da mochila. Curitiba-PR, p. 11p., 2012. Citado na página 15.

LINDEN, R. *Algoritmos genéticos / Ricardo Linden*. 2a. ed. Rio de Janeiro: Brasport, 2008. ISBN: 978-85-7452-373-6, 428 p. Citado 2 vezes nas páginas 15 e 21.

LINDEN, R. *Algoritmos Genéticos*. 3a. ed. Rio de Janeiro: Editora Ciência Moderna Ltda, 2012. ISBN: 978-85-399-205-7, 496 p. Citado 17 vezes nas páginas 15, 16, 17, 19, 21, 22, 23, 25, 26, 27, 28, 29, 31, 32, 45, 47 e 50.

LOBO, E. L. M. *Uma solução do problema de horário escolar via Algoritmo Genético paralelo*. 86f. p. Dissertação (Mestrado) — Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte - MG, fev. 2005. Citado 6 vezes nas páginas 19, 21, 23, 25, 26 e 27.

MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. 3ª. ed. Berlim: Springer Verlag, 1999. Citado na página 22.

MONTGOMERY, D. C.; RUNGER, G. C. Applied statistics and probability for engineers. John Wiley and Sons, 2003. Citado na página 53.

PACHECO, M. A. C. Algoritmos genéticos: princípios e aplicações. Pontifícia Universidade Católica do Rio de Janeiro - ICA: Laboratório de Inteligência Computacional Aplicada, Rio de Janeiro, v. 1, p.9, jul. 1999. Citado na página 20.

REINA, C. D. *Roteirização de veículos com janelas de tempo utilizando Algoritmo Genético*. 90f. p. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2012. Citado 2 vezes nas páginas 14 e 19.

SINGH, V.; SHARMA, S. K.; VAIBHAV, S. Transport aircraft conceptual design optimization using real coded genetic algorithm. *International Journal of Aerospace Engineering*, v. 2016, n. Article ID 2813541, p. 11p., 2016. Citado na página 15.

SOMMERVILLE, I. *Engenharia de Software*. 8ª. ed. São Paulo: Pearson Addison-Wesley, 2007. Citado 2 vezes nas páginas 15 e 17.

SOUSA, J.; CUNHA, M. da C.; MARQUES, A. S. A simulated annealing algorithm for the optimal operation of water distribution networks. *Joint International Conference on Computing and Decision Making in Civil and Building Engineering*, Montréal, Canad, p. 14–16, 10p, June 2006. Citado na página 32.

WANG, J.; ACHARYA, S.; KAM, M. Adaptive decision fusion using genetic algorithm. Department of Electrical and Computer Engineering Drexel University, Philadelphia, Pennsylvania 19104, USA, p. 6p., 2016. Citado 2 vezes nas páginas 22 e 48.

WERNER, J. C.; FOGARTY, T. C. Genetic algorithm applied in clustering datasets. *SCISM, South Bank University*, 103 Borough Road, London, p. 6p., 2015. Citado na página 15.

WIJAYANINGRUM, V. N.; MAHMUDY, W. F. Optimization of ship's route scheduling using genetic algorithm. *Indonesian Journal of Electrical Engineering and Computer Science*, Malang, Indonesia, v. 2, n. 1, p. 180 186, 7p., April 2016. Citado na página 15.

Apêndices

A Projeto do AGSOJ/D

A criação de um projeto de *software* consiste entre outras etapas no conhecimento de Engenharia de Software e a combinação de modelos, técnicas e ferramentas de desenvolvimento de *software* que são indispensáveis para o sucesso do projeto. Cada software deve ser concebido com um conjunto específico de regras de negócios que são chamadas de funcionalidades. As funcionalidades de um software são os requisitos que o mesmo deve possuir para atender as necessidades do negócio. O projeto AGSOJ/D possui as funcionalidades necessárias para resolver o PSOJ/D.

A.0.1 Requisitos Funcionais

Antes de apresentar as descrições dos requisitos funcionais do projeto AGSOJ/D será apresentado em forma de processos a interação do usuário com o sistema evidenciando os principais requisitos funcionais. Esses processos foram criados utilizando a ferramenta CASE Bizagi Modeler 3.0. A Figura 17 apresenta os processos do projeto AGSOJ/D.

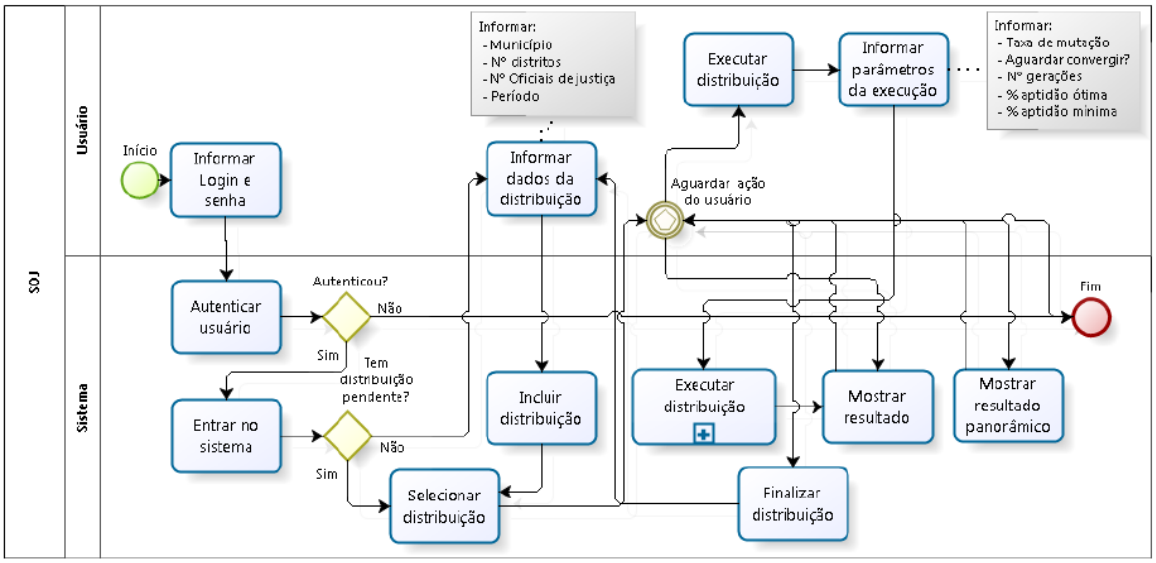

Figura 17 – Processos do projeto AGSOJ/D

A interação inicia-se com o usuário abrindo o sistema e informando o login e senha para autenticação. Se autenticado o sistema permite a entrada do usuário, senão finaliza o sistema. Após a entrada o sistema verifica se existe distribuição pendente para execução, se existir então seleciona a distribuição, senão solicita os dados da distribuição para o usuário que servirá para incluir a distribuição e selecioná-lo. Em seguida a tela principal do sistema é aberta com as seguintes funcionalidades para o usuário: executar distribuição, mostrar resultado, mostrar resultado panorâmico, finalizar distribuição, sair do sistema.

A funcionalidade executar distribuição solicita ao usuário os parâmetros da execução do AGSOJ/D que é submetido ao subprocesso executar distribuição. A Figura 18 mostra os requisitos funcionais do subprocesso executar distribuição.

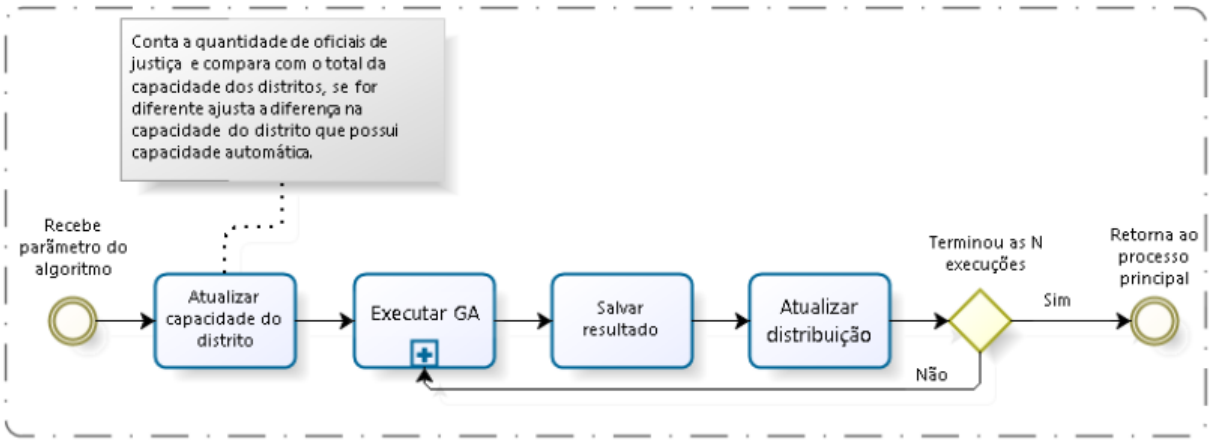

Figura 18 – Subprocesso executar distribuição

As funcionalidades do subprocesso executar distribuição não são visíveis ao usuário, mas são indispensáveis para o funcionamento do sistema. Esse subprocesso inicia-se recebendo os parâmetros do algoritmo informados pelo usuário e atualiza a capacidade de oficiais de justiça do distrito conforme a quantidade de oficiais de justiça que participam da distribuição. Em seguida inicia-se um laço que executa sequencialmente n vezes as seguintes funcionalidades: executar AG, salvar resultado, atualizar distribuição. A Tabela 18 descreve os requisitos funcionais do projeto AGSOJ/D.

A.0.2 Requisitos Não-Funcionais

A Tabela 19 descreve os requisitos não-funcionais do projeto AGSOJ/D.

A.0.3 Principais casos de uso do projeto AGSOJ/D

A Tabela 20 explica o conceito de alguns termos importantes que serão mencionados nos casos de uso.

A seguir são apresentados os principais casos de uso do projeto AGSOJ/D.

[UC001] - Incluir distribuição

Permitir incluir distribuição selecionando o município, informando período, total de distritos e oficiais de justiça.

Pré-condições: Inexistência de distribuição finalizada, se existir distribuição não finalizada o sistema seleciona, senão esse caso de uso é executado.

Pós-condições: Município disponível para distribuição.

Tabela 18 – Requisitos Funcionais

Requisito Funcional	Prioridade	Descrição
RF01 - Autenticar usuário	Importante	Permitir autenticação de usuário.
RF02 - Incluir usuário	Importante	Permitir usuário para acesso ao sistema.
RF03 - Alterar usuário	Importante	Permitir alterar o cadastro do usuário autenticado.
RF04 - Selecionar distribuição	Importante	Funcionalidade não visível ao usuário que seleciona a distribuição pendente para execução.
RF05 - Incluir distribuição	Essencial	Permitir incluir distribuição selecionando o município e período.
RF06 - Incluir distritos	Essencial	Permitir incluir d distritos automaticamente.
RF07 - Incluir oficial de justiça	Essencial	Permitir incluir n oficiais de justiça automaticamente.
RF08 - Executar distribuição	Essencial	Permitir executar a distribuição utilizando AG. Permitir informar antes de realizar a distribuição a quantidade de indivíduos por população, quantidade de gerações para finalizar o AG, selecionar opção para finalizar somente após convergir para o ótimo global. Permitir armazenar os resultados de cada geração com tempo de execução em nano segundos para análise e comparações futuras.
RF09 - Mostrar resultado	Essencial	Permitir imprimir o resultado da distribuição atual por distrito e seus oficiais de justiça.
RF10 - Resultado panorâmico	Essencial	Permitir imprimir o resultado das últimas d distribuições de cada oficial de justiça, comparando com a distribuição atual.
RF10 - Finalizar distribuição	Essencial	Permitir armazenar o resultado da distribuição após finalizar.

Ator: Usuário.

Fluxo de eventos principal

1. O sistema apresenta uma tela contendo os seguintes campos:
   a) Município.* (Selecionará o município: São Luís ou Imperatriz)
   b) Data inicial.*
   c) Data final.*
   d) Total de distritos.*
   e) Total de Oj's.* (Total de oficiais de justiça)
2. O usuário preenche os campos ([FS001] Campos de preenchimento obrigatório);

Tabela 19 – Requisitos Não-Funcionais

Requisito Não-Funcional	Prioridade	Descrição
RNF01 - Usabilidade	Essencial	A interface com o usuário é de vital importância para o sucesso do *software*. Portanto será desenvolvido para Web com as facilidades de navegação desse ambiente.
RNF02 - Desempenho	Importante	Embora não seja um requisito essencial ao sistema, deve ser considerada por corresponder a um fator de qualidade de *software*.
RNF03 - *Hardware* e *Software*	Essencial	Visando criar um protótipo com maior extensibilidade, reusabilidade e flexibilidade, optou-se por adotar como linguagem principal de desenvolvimento a linguagem Java seguindo cuidadosamente as técnicas de orientação a objetos e um *framework* que facilita a implantação dessa solução utilizando outro gerenciador de banco de dados.
RNF04 - Idiomas	Desejável	Permitir internacionalização em Inglês.

Tabela 20 – Termos utilizados nos casos de uso

Termo	Descrição
Ator	Atores representam diferentes tipos de usuários dos sistema, como pessoas, dispositivos de *hardware* ou *software*
Caso de Uso	São narrativas em texto, descrevendo uma unidade funcional, e são amplamente utilizados para descobrir e registrar requisitos funcionais dos sistemas. Os diagramas de casos de uso são representações gráficas dos Casos de Uso e são representados por uma elipse contendo, internamente, o nome do caso de uso.
Símbolo (*)	Indica um campo de preenchimento obrigatório.

3. O usuário clica no botão Incluir;

4. O sistema salva as informações.

Fluxos secundários (alternativos e de exceção)

[FS001] Campos de preenchimento obrigatório

- Se o usuário não preencher algum dos campos obrigatórios da tela, o sistema exibe mensagem solicitando seu preenchimento.

[FS002] Selecionando o município

- Caso tenha selecionado algum, o caso de uso permite incluir.
- Caso contrário, o caso de uso encerra-se aqui.

[UC002] - Executar distribuição

Permitir executar a distribuição utilizando AGSOJ/D com base nos parâmetros de entrada.

Pré-condições: Existência de uma distribuição não finalizado selecionado pelo sistema.

Pós-condições: Existência de distritos e oficiais de justiça para o município selecionado.

Ator: Usuário.

Fluxo de eventos principal

1. O sistema apresenta uma tela contendo os seguintes campos:
   a) Taxa de mutacao.*
   b) Aguardar convergir?.* (aceitar S ou N)
   c) Numero de gerações até convergir.*
   d) Percentual até convergir.*
   e) Percentual mínimo por gene.*
2. O usuário preenche os campos ([FS001] Campos de preenchimento obrigatório);
3. O usuário clica no botão Executar;
4. O sistema realiza n execuções do Algoritmo Genético AGSOJ/D.

Fluxos secundários (alternativos e de exceção)

[FS001] Campos de preenchimento obrigatório

- Se o usuário não preencher algum dos campos obrigatórios da tela, o sistema exibe mensagem solicitando seu preenchimento.

A.0.4 Diagramas de classe

Os modelos conceitual, lógico e físico foram criados utilizando a ferramenta CASE PowerDesigner 16.0. Os diagramas de classe apresentados na Figura 19 representam como são salvos os dados das distribuições finalizadas e que servirão como histórico para a próxima distribuição e fazem parte das regras de negócio do cliente, enquanto que os diagramas de classe apresentados na Figura 20 serviram para armazenar os resultados que foram analisados, comparados e discutidos nesse trabalho. Cada diagrama de classe é composto por um atributo, que representa a chave primária da entidade, e os métodos

específicos das classes. Foram omitidos os demais atributos e os métodos de *getters* e *setters*.

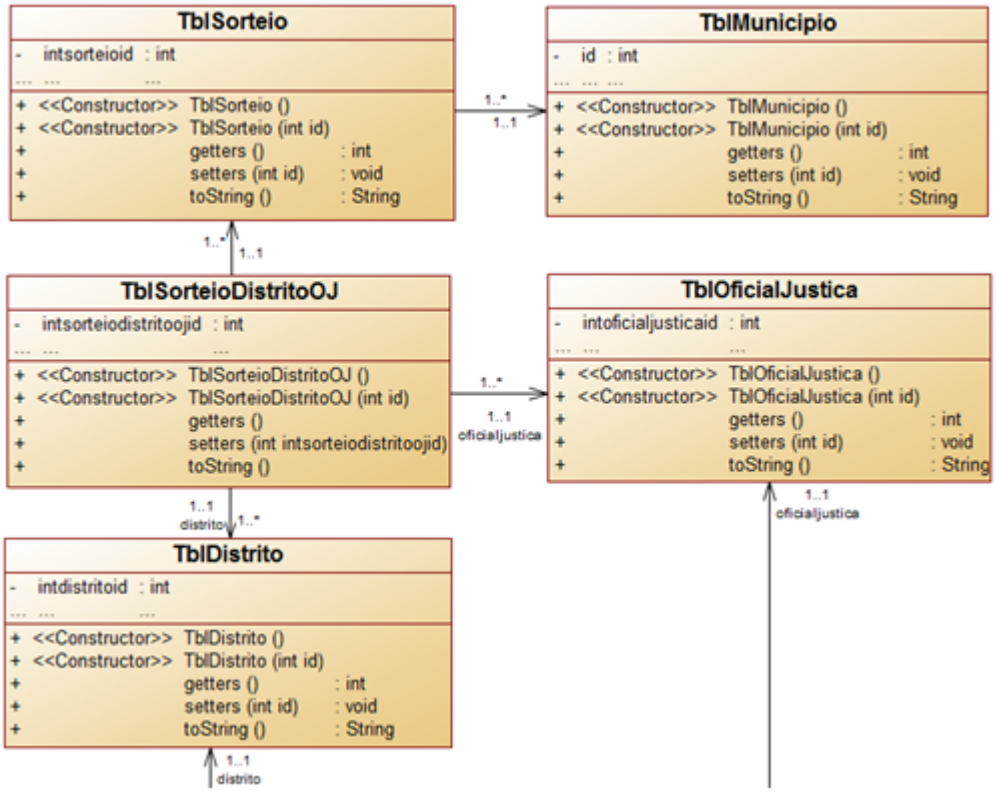

Figura 19 – Diagrama de classe: histórico de distribuições

A classe *TblDistribuicao* relaciona-se com a classe *TblMunicipio* com cardinalidade de 1 para muitos e a *TblDistribuicaoDistritoOJ* que serve para guardar as informações de distritos e oficiais de justiça de determinada distribuição relaciona-se com cardinalidade de 1 para muitos com as classes: *TblDistribuicao*, *TblDistrito* e *TblOficialJustica*.

Os diagramas de classe da Figura 20 por sua vez representam como são salvos os dados da melhor geração após a execução do AGSOJ/D. Esses dados são retornados para serem exibidos como resultado da execução da distribuição e também serviram para criar os gráficos apresentados no capítulo discussões.

A classe *TblGeracaoDistritoOJ*, que guarda as informações da geração-distrito e a aptidão de cada oficial de justiça, relaciona-se com a classe *TblGeracaoDistrito* com cardinalidade de 1 para muitos e, que por sua vez relaciona-se com a classe *TblGeracao* também com cardinalidade de 1 para muitos. Os diagramas de classe da Figura 21 representam a implementação do AGSOJ/D com seus atributos, métodos e relacionamentos.

A classe GA possui o atributo *melhor_geracao* que é uma instância da classe *TblGeracao* e serve para guardar ao longo da execução do AG qual a melhor geração que servirá como retorno do algoritmo. O método *inicializaPopulacao()* cria as instâncias da classe *CromossomoOJ* e guarda na lista população. Os atributos distritos e *cromossomo_oj*

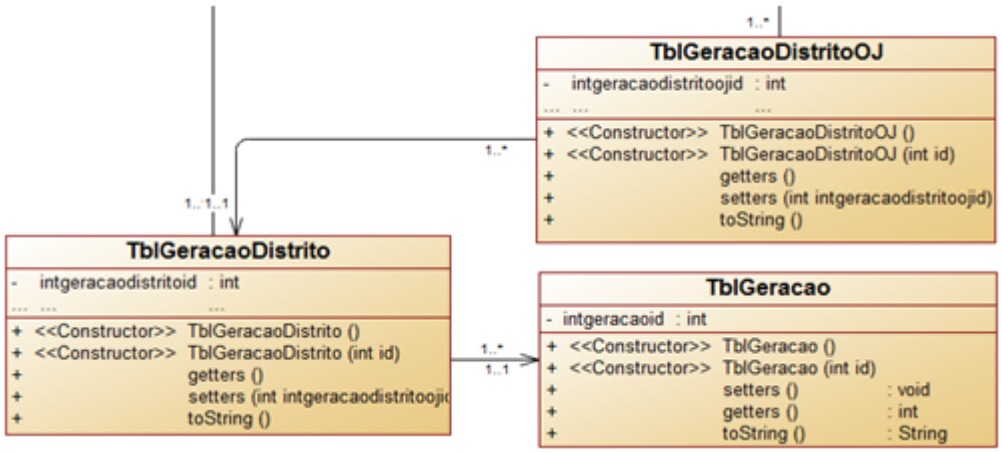

Figura 20 – Diagrama de classe: controle das execuções do AGSOJ/D

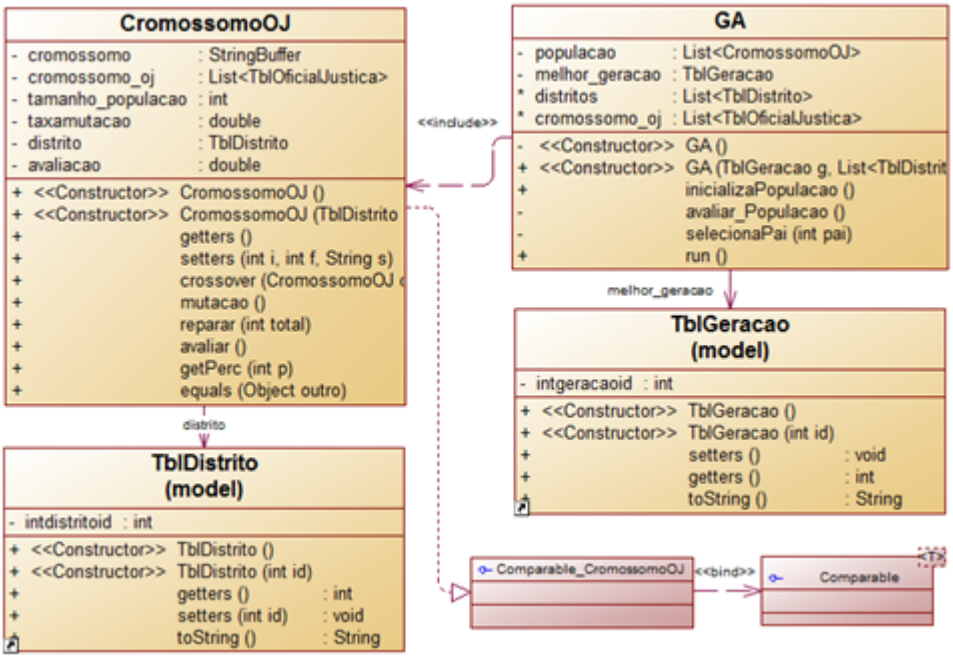

Figura 21 – Diagrama de classe: AGSOJ/D

são parâmetros recebidos pelo AG obtidos da base de dados de distribuições realizadas e finalizadas anteriormente.

A.1 Tecnologias utilizadas

A Tabela 21 contém a lista das tecnologias utilizadas na criação do *software* AGSOJ/D, bem como o link para download.

A técnica utilizada para criação do *software* AGSOJ/D foi a prototipação, que consiste em criar um produto de *software* que irá evoluir até a sua versão final como solução para o PSOJ/D. A prototipação do *software* foi implementada utilizando a ferramenta de

Tabela 21 – Tecnologias utilizadas

Tecnologias	Site para download
brModelo 2.0: Ferramenta de modelagem de banco de dados	http://www.sis4.com/brModelo, acessado em 21 mar. 2016.
Sybase PowerDesigner 16.0: Ferramenta de modelagem e engenharia reversa.	http://powerdesigner.de/en, acessado em 20 maio 2016.
Bizagi Modeler: Ferramenta modelador de processos de negócios.	http://www.bizagi.com/pt, acessado em 20 maio 2016.
PostgreSQL 9.2: Banco de dados	http://www.postgresql.org/download, acessado em 21 mar. 2016.
Eclipse Java EE IDE *for Web Developers*: Ambiente de desenvolvimento integrado	https://eclipse.org/downloads, acessado em 21 mar. 2016.
Framework Mentawai para Java: *Framework* para desenvolvimento Web.	http://www.mentaframework.org, acessado em 17 set. 2015.
HTML5 e CSS3: Recursos avançados para formatação de página Web.	https://www.caelum.com.br/apostila-html-css-javascript, acessado em 28 abr. 2015.

desenvolvimento integrado Eclipse Java EE IDE *for Web Developers*.

Para agilizar o desenvolvimento do protótipo foi necessário estudar utilização do *Framework* Mentawai para Java. O Mentawai é um *framework* MVC *Web* com o objetivo de ser simples e intuitivo, sem o uso de arquivos XML para a sua configuração. Devido à simplicidade é considerado por muitos desenvolvedores o *framework Web* mais fácil de aprender. Em pouco tempo de estudo o iniciante já poderá desenvolver aplicações com bastante recursos.

O banco de dados utilizado no projeto AGSOJ/D foi o PostgreSQL 9.2, por ter além de outras vantagens robustez, segurança e ser um projeto de código aberto.