PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Graduação em Sistemas de Informação

Alexandre Gonzaga Mendes

UTILIZAÇÃO DO ALGORITMO GENÉTICO PARA RESOLUÇÃO DO PROBLEMA DE ALOCAÇÃO DE SALAS

Belo Horizonte
2013

Alexandre Gonzaga Mendes

UTILIZAÇÃO DO ALGORITMO GENÉTICO PARA RESOLUÇÃO DO PROBLEMA DE ALOCAÇÃO DE SALAS

Monografia apresentada ao Instituto de Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: João Caram Adriano

Belo Horizonte
2013

Alexandre Gonzaga Mendes

UTILIZAÇÃO DO ALGORITMO GENÉTICO PARA RESOLUÇÃO DO PROBLEMA DE ALOCAÇÃO DE SALAS

Monografia apresentada ao Instituto de Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

_____

João Caram Adriano
Orientador
Pontifícia Universidade Católica de Minas Gerais

_____

Professor 1
Examinador
Pontifícia Universidade Católica de Minas Gerais

_____

Professor 2
Examinador
Pontifícia Universidade Católica de Minas Gerais

Belo Horizonte, 2013

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus por ter me dado força, coragem e saúde para enfrentar todas as dificuldades encontradas, e colocar ao meu lado pessoas prontas para me ajudar sempre que foi preciso.

Ao meu orientador João Caram pela atenção, boa vontade e pela ótima orientação prestada.

Agradeço aos meus pais e aos meus irmãos por todo apoio, sacrifício e amor dedicados não há palavras para que retribua esse amor.

Aos meus amigos que pelo companheirismo e por toda ajuda prestada.

A empresa GT4W pela confiança, reconhecimento e horas cedidas para que a realização deste trabalho acontecesse e também aos companheiros de trabalho pelo suporte em todos os momentos.

RESUMO

O problema de alocação de salas é um empecilho que ocorre todo inicio de semestre em várias instituições de ensino. Muitas delas ainda resolvem estes problemas de forma manual, o que demanda um grande esforço braçal e demorado que pode levar semanas. Para resolver este tipo de problema de forma otimizada, podemos utilizar as chamadas meta-heurísticas que são técnicas que resolvem os problemas otimização combinatória de forma mais eficiente. As meta-heurísticas dispõem de algoritmos, que são utilizados para encontrar a melhore solução em um tempo viável. Podemos citar como exemplo de algoritmos Busca Tabu, Recozimento Simulado e Algoritmo genético. O caso de teste é aplicado na Universidade Federal de Minas Gerais, em especifico no prédio da Faculdade Filosofia e Ciências Humanas, onde é utilizado o algoritmo genético a fim de resolver o problema descrito que, hoje em dia, é realizado de forma manual. A aplicação do algoritmo genético tem como objetivo tentar gerar uma alocação que atenda o máximo possível à instituição respeitando as restrições da alocação. Testes foram realizados para descobrir o desempenho do algoritmo e definir os melhores parâmetros a serem utilizados, os resultados foram comparados e conclusões foram tomadas de acordo com o que foi obtido durante os testes. Sugestões de possíveis melhorias e continuidade do trabalho são propostas no final.

Palavras-chave: Algoritmo Genético, Problemas de Alocação de Sala, Otimização Combinatória, Meta-Heurísticas.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

AG - Algoritmos Genéticos

BT - Busca tabu

CRUD - *Create, Read, Update, Delete*

CSS - *Cascading Style Sheets*

FAFICH - Faculdade de Filosofia e Ciências Humanas

HTML - *HyperText Markup Language*

MVC - *Model View Controller*

PAS - Problema de Alocação de Salas

POC - Problemas de Otimização Combinatória

PPH - Problemas de Programação de Horários

SGBD - Sistema Gerenciador de Banco de Dados

UFMG - Universidade Federal de Minas Gerais

ÍNDICE

1 INTRODUÇÃO

1.1 Contextualização

Instituições de ensino universitárias se deparam todo início de semestre letivo com o Problema de Alocação de Salas (PAS). Este problema pode ser definido como *Classroom Assignment* que é uma instancia do *course timetabling*, problemas desta instancia são problemas de otimização combinatória. Problemas de otimização combinatória têm a complexidade NP-difícil. Para a resolução de problemas desta complexidade em um tempo razoável são propostas algumas técnicas denominadas meta-heurísticas. Estas técnicas amenizam a dificuldade da resolução destes problemas para encontrar uma solução em um tempo hábil, uma vez que, a resolução destes problemas de forma manual é de grande dificuldade e em alguns casos pode demandar semanas de trabalho da pessoa responsável.

Em suma o trabalho consiste nas distribuições das 149 disciplinas pertencentes aos cursos de graduação e pós-graduação ofertados pela Universidade Federal de Minas Gerais (UFMG), em 50 salas disponibilizadas pelo prédio da Faculdade de Filosofia e Ciências Humanas (FAFICH). Para tentar resolver este problema, será utilizada a meta-heurística chamada de algoritmo genético.

Esta demanda de alocação acontece todo início de semestre e é executada assim que todos os colegiados tenham enviados suas solicitações de necessidades para aquele semestre. O consumo de tempo para realização desta tarefa é de duas semanas podendo se estender quando é necessária a alteração da alocação gerada inicial.

1.2 Objetivos

O tratamento do problema de alocação de salas em instituições de ensino carece de bons trabalhos na literatura. Apesar de se encontrar ferramentas disponíveis, poucas tratam de maneira eficiente as restrições reais existentes nas instituições de forma especifica. Com este trabalho objetiva-se:

1.2.1 Objetivos Gerais

O objetivo deste trabalho é utilizar os conceitos de algoritmo genético para a resolução do problema denominado PAS através do desenvolvimento de um sistema que utiliza algoritmo genético para procura uma solução que atenda todas as necessidades da FAFICH e facilite o gerenciamento das informações da instituição como salas, horários e demais controles das informações.

1.2.2 Objetivos Específicos

Estudo da Implementação do algoritmo genético, em busca de gerar uma alocação que atenda as restrições da universidade e retorne o melhor resultado possível dentro de um tempo adequado.

Otimizar o tempo do gestor através da aplicação do algoritmo genético utilizado para tentar resolver o problema de alocação de salas.

Comparar os resultados que serão realizados nos testes para definir os parâmetros a serem utilizados durante a execução do algoritmo.

1.3 Justificativa

Por mais que existam trabalhos relacionados ao tema as restrições de cada universidade são diferentes. Este trabalho tenta otimizar o tempo do gestor que é responsável por alocar as salas na FAFICH desenvolvendo um sistema que atenda as restrições especificas da instituição.

1.4 Organização do Trabalho

Este trabalho está definido da seguinte forma, foi dividido em seis capítulos, sendo este capítulo 1 e mais cinco. O capítulo 2 apresenta o referencial teórico do trabalho, descrevendo os conceitos utilizados para o desenvolvimento do projeto proposto. No capítulo 3 é apresentada a metodologia e as tecnologias adotadas para desenvolvimento da solução. No capítulo 4 iremos descrever, citar as características e propostas de desenvolvimento e aplicação do algoritmo genético. A conclusão deste trabalho e considerações finais estão apresentadas nos capítulos 5 e 6.

2 REFERENCIAL TEÓRICO

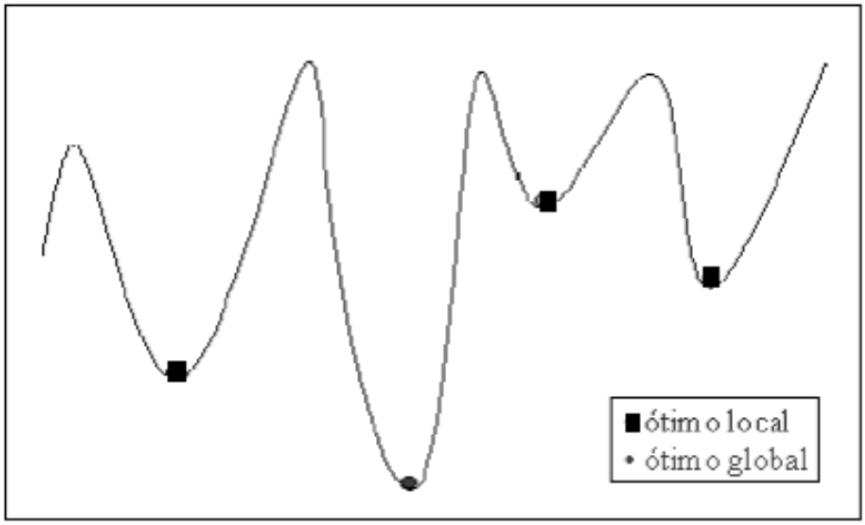
2.1 Problemas de Otimização

Otimização é o processo de encontrar a melhor solução de um problema, também chamada de solução ótima segunda (TIMÓTEO , 2005).

De acordo com (STEIGLITZ; PAPADIMITRIOU, 1982) um problema de otimização é composto pelos termos vizinhança, ótimo local e ótimo global. O termo vizinhança corresponde a um subconjunto do conjunto de soluções do problema. Ótimo local pode ser tratado como o melhor resultado em uma vizinhança, e ótimo global é a melhor solução encontrada no conjunto de acordo com a função objetivo. Função objetivo é uma especificação matemática, retorna como valor a variável que desejamos maximizar ou minimizar. Função objetivo mede o resultado de uma otimização.

De acordo com a figura 1 pode se observar a relação entre ótimo local e ótimo global no conjunto de soluções de um problema típico de minimização, os quadrados mostram soluções quase ótimas chamadas de ótimo local e o circulo mostra a melhor solução encontrada dentro do conjunto de soluções, no caso o ótimo global.

Figura 1 - Representação de um problema de minimização com ótimos locais



Fonte: Timóteo (2005)

Segundo (STEIGLITZ; PAPADIMITRIOU, 1982), os problemas de otimização são divididos em duas categorias: problemas com variáveis contínuas e problemas com variáveis discretas. Os problemas com variáveis discretas também podem ser conhecidos como Problemas de Otimização Combinatória (POC).

Em seu trabalho (RAUPP, 2003) cita que, o problema de otimização combinatória pode ser denominado como a ação de maximizar ou minimizar uma função objetiva de forma a atender restrições, dentro de um contexto.

De acordo com (RAO, 1984) problemas do tipo POC tratam do estudo matemático para encontrar a seleção ótima de objetos discretos, logo, não permitindo, a utilização de métodos clássicos para sua resolução.

Em seu trabalho (GOLBARG; LUNA, 2000) afirmam que a ocorrência de problemas de otimização combinatória pode acontecer em diversas áreas, projetos de sistemas de distribuição de energia elétrica, posicionamento de satélites, roteamento ou escalonamento de veículos, sequenciamento de genes e DNA, classificação de plantas e animais.

De acordo com (CISCON, 2006) em problemas de otimização combinatória, onde existe um grande número de combinações, o que torna inviável a análise de todas soluções possíveis, em um tempo adequado, utilizamos as heurísticas, também conhecidas como algoritmos heurísticos, que são métodos utilizados para encontrar soluções em problemas de otimização combinatória.

2.2 Heurística

O termo heurística é derivado do grego *heuriskein*, o que significa descobrir ou achar. Segundo (STEIGLITZ; PAPADIMITRIOU, 1982), heurísticas são consideradas métodos de aproximação ou métodos de busca de solução. Deve se levar em consideração que não exista uma garantia formal de seu desempenho e uma garantia de que estas heurísticas que irão encontrar uma solução. As heurísticas, apesar de não garantirem encontrar a solução ótima para um problema, procuram por soluções que são consideradas de boa qualidade em um tempo computacional razoável.

Segundo (EVANS; MINIEKA, 1992) heurísticas são necessárias para implementação de problemas NP-Difícil, caso deseje-se resolver tais problemas em um tempo razoável.

As heurísticas contem um grupo com as chamadas meta-heurísticas. Estas meta-heurísticas merecem especial atenção, pois adotam técnicas para amenizar, a dificuldade que os métodos heurísticos têm de escapar dos ótimos locais. As meta-heurísticas podem partir em busca de regiões que possam conter o melhor resultado

no conjunto de soluções, alem disto, as meta-heurísticas podem ser aplicadas à maioria dos problemas de otimização combinatória (NASCIMENTO; SILVA; ALVARENGA , 2005) .

Segundo (OLIVEIRA, 2006) uma heurística é a instanciação de uma meta-heurística, ou seja, a aplicação da mesma em um problema específico de otimização.

Como exemplos de meta-heurísticas temos Busca Tabu (*Tabu Search*), Otimização por Colônias de Formigas (*Ant Colony Optimization*), Recozimento Simulado (*Simulated Annealing*) e Algoritmo Genético (*Genetic Algorithm*).

2.3 Busca Tabu

Busca tabu (BT) é uma meta-heurística adaptativa, que utiliza uma estrutura de memória através de uma lista, contendo um histórico das evoluções para evitar que o processo de busca forme ciclos, ou seja, retorne a um local ótimo visitado anteriormente (SOUZA, 2000), (AR-MENTANO; BRANCHINI, 2013) e (SUBRAMANIAN et al. , 2006).

Segundo (SUBRAMANIAN et al., 2006) BT é uma técnica que foi desenvolvida por (GLOVER, 1986) com o objetivo de encontrar soluções para problemas de programação linear. Ao formalizar a técnica, o autor publicou uma série de trabalhos envolvendo diversas aplicações da meta-heurística.

Basicamente o funcionamento do BT a feito partir da definição de uma população inicial “S”, o algoritmo explora cada iteração de um subconjunto V que representa a vizinhança N(S) da solução “S”. O membro S’ de V com melhor valor nessa região segundo a função f(.) torna-se a nova solução mesmo que S’ seja pior que S isto é, que f(S’) > f(S) (SOUZA , 2000) . O anexo A representa o pseudocódigo do algoritmo da Busca Tabu.

Segundo (ARMENTANO; BRANCHINI, 2013) o algoritmo tem um intensivo uso de memória o que é essencial para está técnica já que trabalha com listas na memória. Para os autores o uso da memória faz com que a busca em regiões com tenha grande chances de encontrar o resultado, ou até mesmo, diversificar a busca através de regiões inexploradas.

Ainda de acordo com (ARMENTANO; BRANCHINI, 2013) algumas técnicas devem ser adotadas para que o algoritmo tenha um melhor resultado: lista tabu dinâmicas, passagens por regiões planas, intensificação, diversificação, *path relinking*. Lista tabu dinâmicas tem como objetivo evitar que o algoritmo entre em processo de ciclo. Passagens por regiões planas podem levar o algoritmo a pensar que não existem melhoras significativas na qualidade das soluções e atingir o critério de parada. Para evitar esta situação é necessário aumentar o tamanho da lista tabu enquanto o algoritmo estiver passando pela região plana e voltar a reduzir quando houver mudança no valor da função objetiva. Intensificação é uma técnica utilizada para concentrar os esforços da pesquisa em regiões consideradas promissoras. Diversificação é uma técnica que utiliza memória de longo prazo para redirecionar a pesquisa para regiões que ainda não foram suficientemente exploradas. *PathRelinking* trada da intensificação de incorporar atributos de soluções de boa qualidade, em seguida explora caminhos que contenham uma ou mais soluções de elite.

2.4 Recozimento Simulado

Recozimento Simulado é a técnica de busca local probabilística, proposta por (KIRKPATRICK; JR.; VECCHI, 1983), com base nos fundamentos da termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos.

Conforme (NORONHA, 2003) cita quando levamos um cristal a sua temperatura de fusão, as moléculas ficam desordenadas e se agitam livremente. Ao resfriar-se a amostra de maneira infinitamente lenta, as moléculas vão adquirir a estrutura cristalina estável que tem um nível de energia mais baixo.

Segundo (SOUZA; MARTINS; ARAÚJO, 2002) o processo se inicia com uma solução, normalmente gerado aleatoriamente, e então é selecionado um de seus vizinhos de forma randômica. Se este vizinho for melhor que o original ele é aceito e substitui a solução atual. Se ele for pior por uma quantidade Δ, ele é aceito com uma probabilidade e então é aplicada está formula $-\Delta/T$, onde T decresce gradualmente conforme o andamento do algoritmo. Esse processo é repetido até que T seja tão pequeno que mais nenhum movimento seja aceito. A melhor solução encontrada durante a busca é pode ser aceita como a solução ótima. A meta-heurística *Simulated Annealing* é derivada de simulações em termodinâmica e por isto o

parâmetro T é referenciado como temperatura e o processo de reduzir o seu valor é chamado de processo de resfriamento. O apêndice B representa o pseudocódigo do algoritmo *Simulated Annealing*.

2.5 Algoritmos Genéticos

De acordo com (GOLDBERG, 1989) Algoritmos Genéticos (AG) são algoritmos baseados na teoria da evolução das espécies criada por (DARWIN, 1968) utilizando os conceitos da biologia genes, indivíduo, população, cromossomos, cruzamento, mutação e seleção. Estes algoritmos foram introduzidos por (HOLLAND, 1975) para resolver os problemas chamados *timetabling* descritos na próxima sessão.

Para entender melhor (MITCHELL, 1998) descreve os termos biológicos necessários para o funcionamento dos algoritmos genéticos. Gene se trada de uma característica particular de um cromossomo. Um cromossomo é composto por um ou mais genes, pode se dizer também que o cromossomo é uma seqüência de genes que será considerada uma solução do problema. *Fitness* significa a aptidão do indivíduo em um determinado ambiente. Individuo é a combinação do cromossomo mais o *fitness* que é calculado através de uma função objetivo. População é um grupo de indivíduos. Geração se trata de cada interação do algoritmo.

Em seu trabalho (LUCAS, 2000) descreve algoritmos genéticos da seguinte forma: São algoritmos que trabalham sobre uma população, através de uma função de adaptação *fitness*, para que aconteça a evolução. Primeiramente é inicializada uma população, logo após os operadores genéticos de seleção, reprodução também conhecida como *crossover* e mutação podem ocorrer, o processo se repete até que os critérios de parada aconteçam. Também afirma que os termos utilizados pertencem à tradição existente no meio da computação evolutiva de utilizar, com certa liberdade os termos da biologia.

Segundo (OLIVEIRA, 2005), o processo de evolução executado por um algoritmo genético corresponde a de busca uma solução que possa atender ao objetivo do problem.

Figura 2 - Estrutura funcional de um algoritmo genético típico

Seja $S(t)$ a população de cromossomos na geração t.

$t \leftarrow 0$
inicializar $S(t)$
avaliar $S(t)$
enquanto o critério de parada não for satisfeito **faça**
    $t \leftarrow t+1$
    selecionar $S(t)$ a partir de $S(t-1)$
    aplicar *crossover* sobre $S(t)$
    aplicar mutação sobre $S(t)$
    avaliar $S(t)$
fim enquanto

Fonte: (LACERDA; CARVALHO , 1999)

A Figura 2 trata de uma estrutura funcional típica de um algoritmo genético. Segundo (LACERDA; CARVALHO, 1999) citam, o primeiro passo a ser tomado é a geração de uma população inicial, que é formada através de métodos aleatórios para gerar os indivíduos, assim teremos uma biodiversidade na população. Durante o processo evolutivo todos os indivíduos da população são avaliados e cada um deles recebe o seu *fitness*, o que representa a qualidade da solução representada por ele.

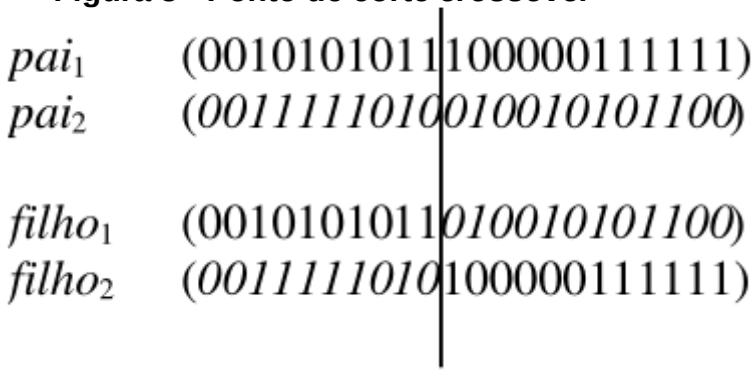
Segundo (LOBO, 2005) existem vários métodos para selecionar indivíduos para execução dos operadores genéticos, em seu trabalho o autor apresenta os seguintes métodos. Seleção por roleta que é um método tradicional onde para cada indivíduo é atribuído um espaço na roleta, sendo o tamanho proporcional ao valor do *fitness* do individuo. Esta roleta gira N vezes onde N é o numero do tamanho da população, selecionando assim os pais para próxima geração. A Seleção por torneio inicialmente seleciona X indivíduos da população onde X é um valor aleatório anteriormente escolhido, logo em seguida são selecionado os dois indivíduos que contêm o maior valor de *fitness* no caso os país. Segundo o autor o método de seleção torneio é o mais utilizado, pois oferece a vantagem de não exigir que a comparação seja feita entre todos os indivíduos da população.

De acordo com (GÓES, 2005) os principais operadores genéticos, utilizados ao se desenvolver algoritmos genéticos são mutação e *crossover*. Estes são responsáveis por realizar transformações nos indivíduos da população porem, cada um possui diferentes funções dentro do algoritmo.

Mutação é um operador que modifica a genética de um gene ele é fundamental para garantir a biodiversidade da população, ainda segundo (GÓES, 2005) vários autores afirmam que é o operador mais importante e optam por trabalhar somente com este operador. Este operador é fundamental para o desenvolvimento de algoritmos genéticos por evitar a convergência prematura da solução, ou seja, quando uma população se estabiliza com um *fitness* pouco adequado, podemos dizer então que, um super-indivíduo domina o processo seletivo de tal forma que não é possível gerar filhos melhores, este mesmo super-indivíduo transmite suas características para toda a população.

O operador cruzamento também conhecido como *crossover* é um operador genético que utiliza um ponto de corte para dividir os indivíduos e produzir duas cabeças e duas caldas, neste caso são os genes dos pais escolhidos no método de seleção. Após isto é realizada a troca das caldas dos pais criando dois filhos contendo material genético similares aos dos pais a figura 3 mostra onde é realizado o ponto de corte como ficam os filhos criados após a troca das caldas (LACERDA; CARVALHO , 1999).

Figura 3 - Ponto de corte *crossover*



Fonte: (LACERDA; CARVALHO , 1999)

No artigo apresentado pelos autores (LACERDA; CARVALHO, 1999) o elitismo é descrito como um operador genético que foi proposto por (JONG, 1975) em um de seus trabalhos que é um dos pioneiros no assunto, algoritmos genéticos. Uma vez que os melhores indivíduos, de acordo com a função objetiva, podem ser perdidos entre uma geração e outra devido a execução dos operadores genéticos *crossover* e a mutação, torna-se viável transferir o melhor indivíduo para a próxima geração, o nome dado para está estratégia é elitismo uma técnica muito utilizada ao se desenvolver AGs. Os autores apresentam gráficos que mostram o desempenho da utilização do operador, segundo os mesmo quando se é utilizado o operador fica

claramente observado que, com o uso do elitismo, o algoritmo encontra a solução mais rápida, do que quando não ocorre a utilização do operador.

Segundo (HAMAWAKI, 2011) e (OLIVEIRA, 2005) algoritmos genéticos são eficientes para encontrar soluções ótimas ou quase ótimas, pois as limitações são mínimas dos demais métodos de busca tradicionais. Ainda segundo (OLIVEIRA, 2005), os algoritmos genéticos têm se mostrado ferramentas poderosas para resolver problemas onde o espaço de busca é muito grande e os métodos convencionais se mostraram ineficientes.

2.6 TimeTabling

Segundo (LUVEZUTE; KRIPKA, 2013) os Problemas de Programação de Horários (PPH), também conhecidos como *TimeTabling*, são os problemas que mais se destacam nas organizações acadêmicas. De acrodo com (SCHAERF, 1999) estes problemas são divididos em três categorias *school timetabling, course timetabling* e *examination timetabling.*

School TimeTabling: Trata-se basicamente da geração de horários semanais, em escolas de segundo grau, onde deve-se evitar os choques entre os horários das disciplinas e que cada professor receba apenas uma turma para cada horário. Neste caso o aluno recebe um número fixo de disciplinas a serem cursadas.

Course TimeTabling: Diz respeito à alocação de aulas de uma universidade típica. Neste problema os alunos podem escolher as matérias em que vão se matricular, portanto o problema tem como objetivo minimizar os possíveis choques entre as disciplinas, professores e horários disponibilizados pela instituição de ensino.

Examination TimeTabling: Aborda o problema de programação de horários dos exames da instituição, de maneira que, disciplinas que tenham alunos em comum, distanciem o máximo possível as datas dos exames.

Segundo (PINHEIRO; OLIVEIRA, 2001) o problema de programação de horários vem sendo abordado desde a década de 60, sendo que os primeiros trabalhos a se destacarem foram realizados na década de 80.

O Problema de Alocação de Salas (PAS) também conhecido como *Classroom Assignment* é tratado como parte do problema de programação de cursos universitários *course timetabling*. Conforme cita (MARINHO, 2005) várias instituições

universitárias se deparam com o PAS durante o início de cada semestre letivo. Este problema é considerado NP-Difícil por (EVEN; ITAI; SHAMIR, 1975) e (CARTER; TOVEY , 1992), com isto, a determinação da solução ótima do problema, em um período de tempo aceitável se torna uma tarefa difícil.

Uma vez que é de extrema dificuldade encontrar a solução ótima do PAS em tempo razoável, este problema é normalmente tratado através de técnicas heurísticas, que apesar de não garantirem encontrar a solução ótima do problema, são capazes de retornar uma solução de qualidade em um tempo adequado (NASCIMENTO; SILVA; ALVARENGA, 2005). Segundo (EVEN; ITAI; SHAMIR, 1975) o PAS pertence à classe de Problemas de Otimização Combinatória (POC).

Segundo (LUVEZUTE; KRIPKA, 2013) o problema deve considerar que as disciplinas dos cursos universitários já tenham seus horários de início e de término definidos. O problema se resume então na alocação das disciplinas às salas desta universidade respeitando os horários destas disciplinas e as demais restrições exigidas.

Em seu trabalho (SOUZA, 2000) afirma que boa parte das universidades ainda resolve este problema de forma manual, o que torna o processo árduo e demorado, podendo levar vários dias para ser concluído.

2.7 Trabalhos Relacionados

Foram encontrados vários trabalhos relacionados ao tema de resolução de problemas de otimização combinatória através de meta-heurísticas. Os trabalhos relacionados escolhidos contêm formas diferentes de resolução do PAS.

Ao desenvolver seu trabalho (SUBRAMANIAN et al., 2006) concluirão que o BT teve um resultado adequado e sempre produzindo melhorias durante o processo de refinamento da solução inicial, os mesmos também afirmam que o algoritmo apresentou-se robusto, uma vez que não houve grande variação na solução final, portando o mesmo sempre gerava soluções satisfatórias.

A abordagem realizada por (SILVA, 2005) foi através da meta-heurística recozimento simulado. A autora afirma que o algoritmo se mostrou eficiente para resolução do PAS, produzindo bons resultados ao atender a maioria dos requisitos do problema, o uso deste algoritmo é indicado quando se trada da substituição do processo manual realizado pelas instituições de ensino.

O trabalho desenvolvido por (HAMAWAKI, 2011) é realizado através da utilização de algoritmos genéticos, segundo a autora a aplicação do AG e a analise dos resultados obtidos é possível concluir que as técnicas implementadas favorecem a obtenção de uma solução satisfatória e eficiente.

A resolução do problema PAS através de diferentes técnicas meta-heurísticas, podemos notar que nas conclusões dos trabalhos apresentados uma solução satisfatória pode ser obtida quando se utiliza estas técnicas através dos algoritmos.

3 METODOLOGIA

A resolução deste trabalho está dividida nos seguintes tópicos análise, criação do ambiente e desenvolvimento do sistema não necessariamente nesta ordem.

O primeiro passo foi à definição da utilização do algoritmo genético, por ser um algoritmo muito utilizado no meio acadêmico para solucionar este tipo de problema. Para composição do algoritmo foram implementadas as seguintes funcionalidades: criação da população inicial, método de crossover, método de mutação, método de elitismo, método de seleção por roleta e o fluxo completo do algoritmo.

Logo em seguida realizada foi realizada uma entrevista com o responsável pela resolução do problema PAS na UFMG. As perguntas utilizadas nesta entrevista podem ser obtidas no anexo A. Os seguintes diagramas foram desenhados: diagrama de caso de uso, diagrama de classe e diagrama de entidade relacionamento. Estes foram escolhidos para que o sistema tenha uma documentação mínima tendo em vista que o foco do trabalho é a resolução do problema PAS através da utilização do algoritmo genético.

Após a escolha do algoritmo, entrevista com o gestor e desenho dos diagramas a utilização de cada um dos itens descritos acima foram interpretados de acordo com o problema em forma de algoritmo genético, estes passos são descritos no capítulo 4 com mais detalhes.

Em seguida foi realizada uma série de dados para definir os melhores parâmetros iniciais do algoritmo de forma que ele atenda todas as restrições do problema e tenha o menor tempo possível de execução até que o algoritmo encontre a melhor solução possível. Após a realização dos testes a análise a conclusão dos dados foi obtida e é apresentada no capitulo 5 detalhadamente.

3.1 Ferramentas Utilizadas

A linguagem para o desenvolvimento do algoritmo escolhida foi o Java. O mesmo teve sua criação pela Sun Microsystems e utiliza o conceito de máquina virtual. É uma linguagem *open-source* e tem suporte a orientação a objetos (CAELUM, 2013).

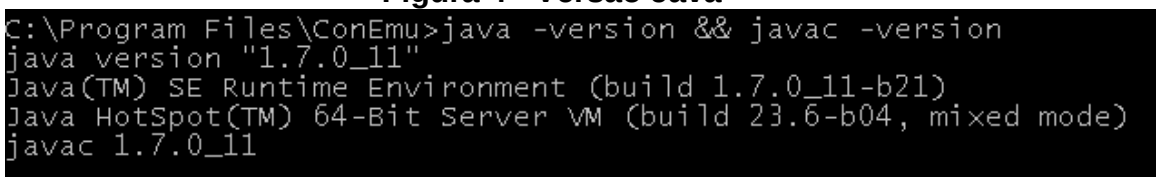
Foi o utilizado *PostgreSQL* um sistema gerenciar de banco de dados objeto-relacional de código aberto para persistir o dados da aplicação. Este Sistema Gerenciador de Banco de Dados (SGBD) têm suporte ao uso de chaves estrangeiras, *Joins*, *Triggers e procedures*, além de suporte de integração ao Java (POSTGRESQL, 2013).

Para produtividade do desenvolvimento foi escolhido um *framework* MVC (*Model View Controller*) o PLAY! que é um *framework open-source,* utiliza Java e *hibernate* em sua arquitetura. A versão utilizada deste *framework* para a realização do trabalho foi a 1.2.5 (PLAY!, 2013).

Para o desenvolvimento da interface foi escolhida a linguagem de marcação HTML (*HyperText Markup Language*) que é uma linguagem *open-source* utilizada no desenvolvimento de paginas web (W3C, 2013 a). CSS (*Cascading Style Sheets*) também foi utilizado este recurso é responsável por adicionar estilos em documentos escritos em linguagem de marcação (W3C, 2013 b). Também foi utilizado o *JavaScript* que é uma linguagem de script *client-side* utilizada para adicionar funcionalidades nas páginas HTML e para se comunicar com os *webServers (W3SCHOOLS , 2013).*

Antes da implementação do sistema demovemos realizar a configuração do ambiente para o início do desenvolvimento os seguintes passos devem ser tomados para que o ambiente seja reproduzido novamente. Primeiramente deve se instalar o *PostgreSQL*, logo em seguida deve se instalar o Java 7 em sua maquina, para validar a instalação deve-se executar o seguinte comando "*java -version && javac -version*" a mensagem descrita na figura 4 deve ser mostrada.

Figura 4 - Versão Java



Fonte: Elaborada pelo autor

Após a instalação do Java deve-se instalar o framework *Play*! seguindo os passos encontrados no site do fabricante para validar a instalação do *Play*! deve se executar o comando "*play version*" e o *prompt* de comando deve retornar a mensagem conforme mostra a figura 5.

Figura 5 - Versão Framework

Fonte: Elaborada pelo autor

Uma vez que o ambiente já está configurado devemos iniciar um projeto no framework instalado através do comando "*play new nomeDoProjeto*".

Foi criado um *script* para criação do banco de dados e população inicial com as informações sobre prédios, salas, turnos, horários.

4 SISTEMA DESENVOLVIDO

Todo início de semestre, instituições de ensino se deparam com o PAS, um problema de alocação de salas que tem sua complexidade definia como NP-Difícil. Logo a resolução manual de problemas desse tipo se torna inviável em um espaço de tempo curto. Para resolver estes problemas são utilizadas as chamadas meta-heurísticas que são métodos de aproximação, que ajudam a encontrar uma solução ótima em um conjunto de soluções viáveis, podemos citar como exemplos de meta-heurísticas os seguintes algoritmos: Busca Tabu, Algoritmo Genético e Recozimento Simulado.

Atualmente o trabalho desenvolvido, pelo gestor responsável por alocar as disciplinas nas salas no prédio da FAFICH, consome o tempo de duas semanas para gerar a primeira versão de alocação. Porem algumas alterações são feita nas restrições de horários das disciplinas, neste caso é necessário readaptar a alocação que já estava pronta, gerando retrabalho e um consumo maior de tempo para o envio da versão final da alocação.

Para este trabalho foram utilizadas as informações fornecidas pela FAFICH que correspondem ao primeiro semestre de 2013. O prédio a ser alocado tem 50 salas disponíveis e durante este semestre tiveram o total de cinco cursos, cada um com dois colegiados e o total de 149 disciplinas com suas restrições de horário especificas. Para a resolução do problema devem-se verificar as seguintes restrições durante a alocação: a quantidade de vagas da disciplina com a quantidade de vagas da sala, os horários em que a disciplina deve ser alocada tem que coincidir com o horário da sala em que foi alocada, e todos os horários específicos de uma determinada disciplina devem estar na mesma sala. Estas informações foram coletadas com o gestor da área através das perguntas descritas no anexo A.

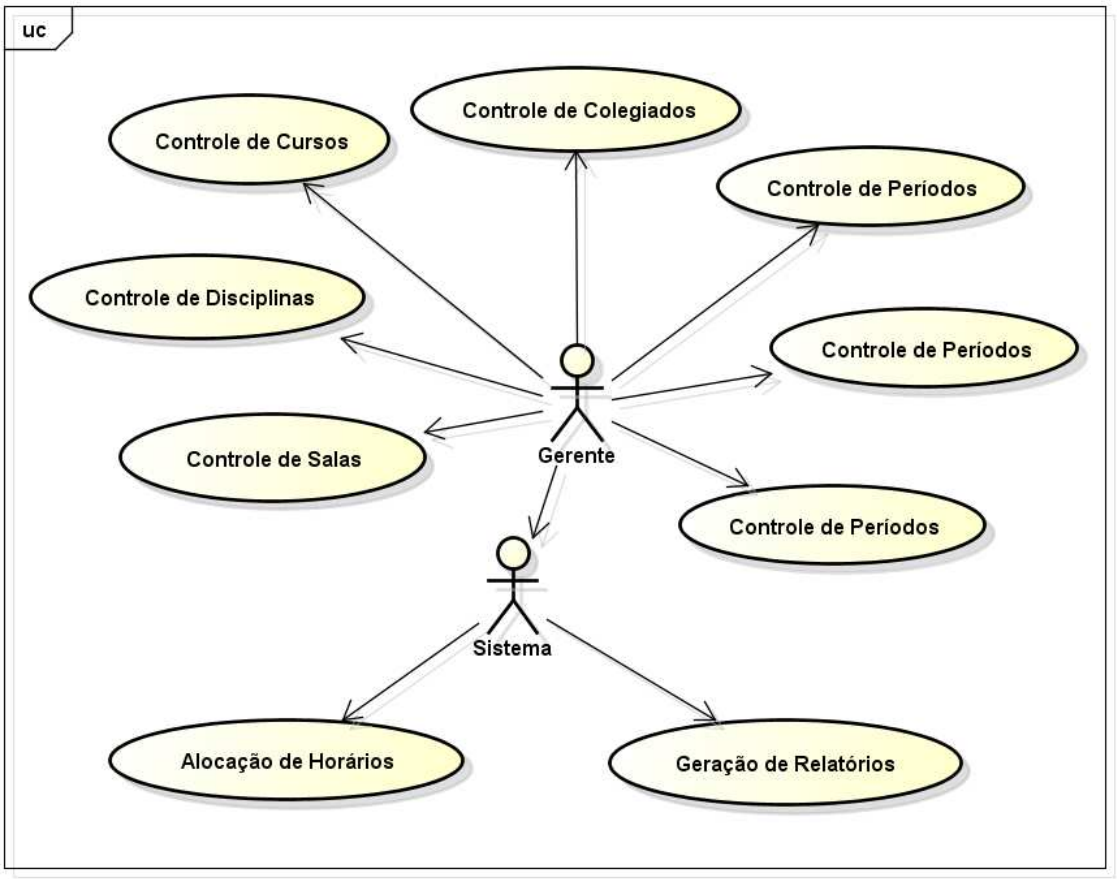
A resolução do problema consiste então, na alocação dos horários das disciplinas respeitando as restrições do problema. A técnica utilizada para tentar solucionar o problema é chamada de algoritmo genético uma meta-heurística bastante utilizada na resolução de problemas deste tipo.

4.1 Modelagem

4.1.1 Diagrama de Caso de Uso

A figura 6 descreve todas as funcionalidades que o sistema possui. Essas funcionalidades foram dividias em dois atores, "Gerente" e "Sistema", cada ator ligado com suas respectivas funcionalidades, porém, o "Gerente" pode acessar o ator "Sistema" para ter acessos funcionalidades que são encontradas no mesmo. O sentido das setas informa o que cada ator pode acessar no sistema.

Figura 6 - Diagrama de Caso de Uso



Fonte: Elaborada pelo autor

As funcionalidades "controle de turnos", "controle de horários", "controle de prédios", "controle de salas", "controle de cursos", "controle de colegiados", "controle de períodos" e "controle de disciplinas" são disponibilizadas através de módulos com as funcionalidades *Create, Read, Update, Delete* (CRUD) para que o responsável

tenha total controle sobre as informações a serem administradas. As telas do sistema podem ser encontradas no anexo B

Já as funcionalidades "alocação de horários" e "geração de relatórios", são rotinas executadas pelo "Sistema". A funcionalidade alocação de horários é uma rotina que utiliza conceitos de algoritmo genético para encontrar a melhor solução do problema e a funcionalidade geração de relatórios mostra para o "Gerente" o melhor resultado de alocação encontrada pelo algoritmo. Detalhes do funcionamento serão explicados nas seções posteriores.

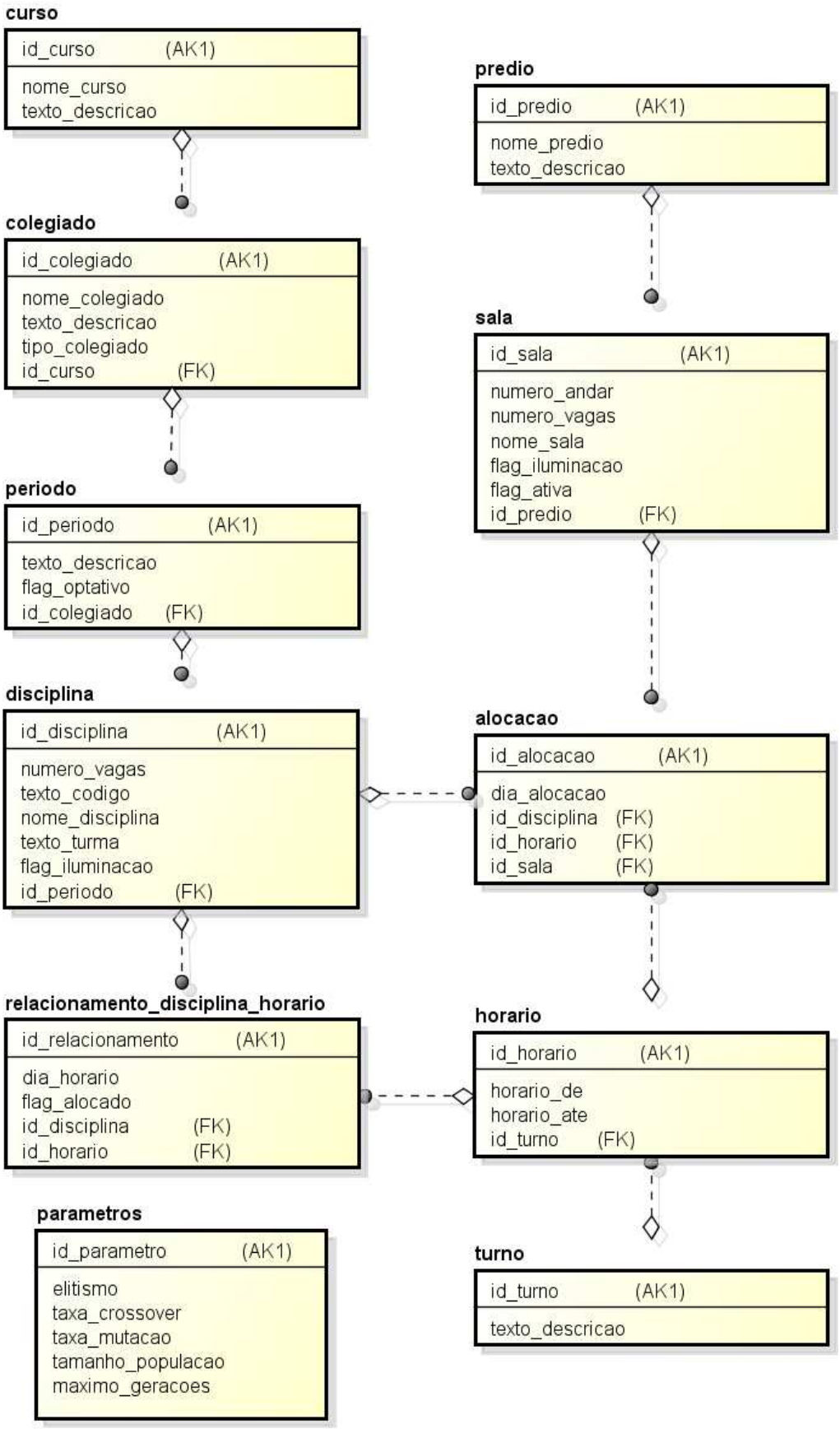
4.1.2 Diagrama de Entidade Relacionamento

A Figura 7 mostra o relacionamento das tabelas no sistema através do diagrama de entidade relacionamento. As tabelas curso, colegiado, período, disciplina, sala, prédio, turno, horário são utilizadas para armazenar as informações dos respectivos objetos que são controlados pelos módulos de CRUD e suas respectivas funcionalidades.

A tabela relacionamento disciplina horário, salva as obrigatoriedades dos horários das disciplinas, uma vez que, todos os horários são montados pelos colegiados e não pelo responsável pela alocação das disciplinas nas salas.

A tabela alocação contém o relacionamento de disciplina, horário e sala o que corresponde a melhor alocação encontrada pelo algoritmo. O item disciplina pode ter o seu valor nulo.

Figura 7 - Diagrama de Entidade Relacionamento

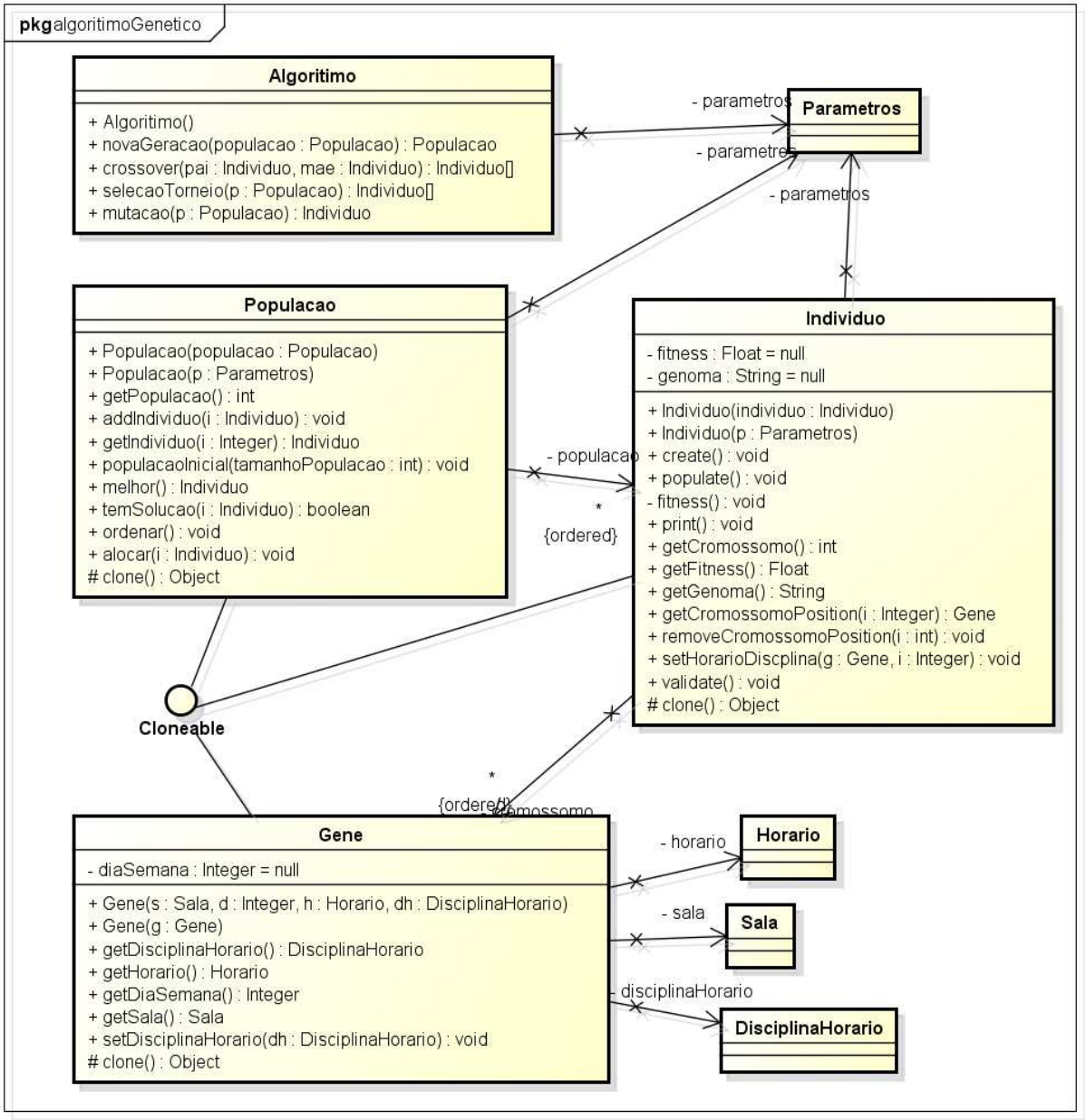


Fonte: Elaborada pelo autor

4.1.3 Diagramas de Classe

O diagrama de classe apresentado na figura 8 mostra as classes utilizadas durante a execução do algoritmo genético, mais detalhes sobre o funcionamento do algoritmo serão explicados na sessão 4.2.

Figura 8 - Diagrama de Classe



Fonte: Elaborada pelo autor

4.2 Algoritmo Genético

4.2.1 Indivíduo

O termo gene representado pela figura 9 possui uma combinação de quatro variáveis: sala, dia da semana, horário e o relacionamento de obrigatoriedade "disciplina horário", onde é verificada a restrição de horário daquela disciplina. As variáveis, sala, horário e dia da semana são fixas e não podem ser nulas uma vez que todas as combinações possíveis destas três variáveis formam um cromossomo que tem um tamanho fixo para todos os indivíduos.

Um gene com o relacionamento, "disciplina horário" igual a nulo representa um horário vago para aquela combinação especifica de sala, horário e dia da semana.

Figura 9 - Representação Gene

Gene			
Sala	Dia da Semana	Horário	Relacionamento
2	3	4	2

Gene			
Sala	Dia da Semana	Horário	Relacionamento
2	3	5	null

Fonte: Elaborada pelo autor

Cromossomo é uma sequência de genes, esta sequência tem um tamanho fixo e pode ser medido pela seguinte fórmula.

*(número de Salas * número de Horários * número de dias da semana)*

Inicialmente o individuo contem todas as variáveis, “relacionamento de obrigatoriedade” com o valor nulo, estas variáveis serão preenchidas randomicamente na criação da população inicial que em breve será explicada. O cromossomo do indivíduo preenchido com os relacionamentos de obrigatoriedades representa uma alocação.

Uma alocação que possa atender as restrições do problema, é medida pela pontuação de *fitness* que é calculada pela função objetivo, está alocação pode ser ou não o resultado ótimo global do problema. Os valores utilizados na representação do cromossomo na figura 10 são os *Id's* do “relacionamento de obrigatoriedade” entre horário e disciplina. Em caso de horário vago em um determinado gene esta variável terá o valor nulo.

Figura 10 - Representação Cromossomo

Cromossomo							
1	null	null	3	4	null	...	7

Fonte: Elaborada pelo autor

O termo indivíduo é definido neste trabalho pela composição do cromossomo juntamente com a pontuação adquirida após a execução do método calculo de *fitness*, a figura 11 demonstra a representação de um indivíduo.

Figura 11 - Representação Indivíduo

Individuo								
Fitness	Cromossomo							
12.0	1	null	null	3	4	null	...	7

Fonte: Elaborada pelo autor

4.2.2 Definição da Função Objetivo

O calculo da função objetivo é realizado através da execução da fórmula apresentada a seguir, está fórmula é utilizada para calcular cada restrição do problema.

A primeira restrição chamada de *fitness01 trata* da restrição de obrigatoriedade entre as disciplinas e horários, onde uma determinada disciplina deve ter todos os seus horários atendidos de acordo com o relacionamento de obrigatoriedade. É realizado um somatório de todos “relacionamentos de obrigatoriedade” que são atendidos de acordo com os dados contidos no gene em que está alocado, em seguida utilizando o somatório dos relacionamentos contidos no problema que estão em sua posição correta é aplicada a regra de três.

*fitness01 = somatórioRelaciomentosAtendidos * 100 / quantidadDisciplinaHorario*

O mesmo calculo é realizado para as outras restrições, utilizando o somatório das disciplinas que tem os horários estão na mesma sala chamamos de *fitness02*, e disciplinas que a capacidade da sala é atendida *fitness03*.

O *fitness* final é calculado pela fórmula abaixo o resultado, tem como valor máximo de 100 e mínimo de 0.

(fitness01+fitness02+fitness03)/3

Estás fórmulas foram utilizadas, pelo fato de maximizarem a função e retornarem como resultado uma porcentagem que definirá quanto atende a solução proposta pelo indivíduo.

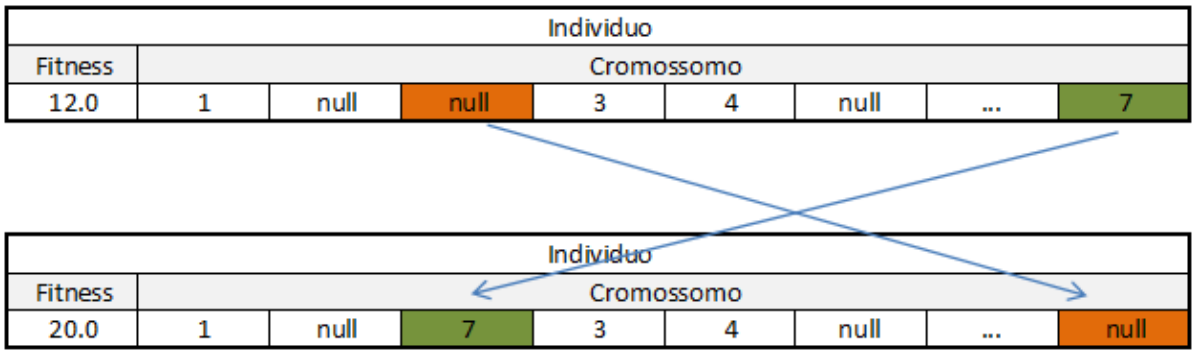
4.2.3 Operadores Genéticos

Um dos operadores genéticos utilizados neste trabalho é o elitismo. Este operador genético com valor verdadeiro garante que o melhor indivíduo sempre vá para a próxima geração.

Mutação é a inversão genética dos genes, de um indivíduo escolhido randomicamente, da população anterior. Após a realização da mutação genética o indivíduo é inserido na nova população.

Primeiramente são escolhidos de forma randômica dois genes do cromossomo. Após a escolha dos genes que serão trocados, é feita a troca dos mesmos de posição. No caso apenas a relação de obrigatoriedade entre a disciplina e horário é trocada, permanecendo as outras propriedades. Após a mutação este indivíduo recebe uma nova nota de *fitness* de acordo com a sua nova sequencia de genes. Está nota pode ser maior ou menor do que a anterior, a figura 12 representa o processo descrito.

Figura 12 - Representação Mutação

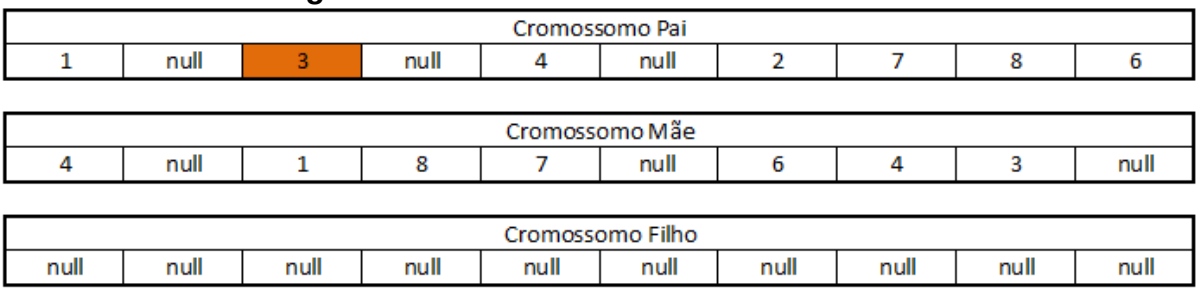


Fonte: Elaborada pelo autor

O método seleção por torneio é utilizado para selecionar os indivíduos que iram participar do *crossover*. O método escolhe três indivíduos da população anterior randomicamente e dos três indivíduos escolhidos, dois que contêm a maior pontuação de *fitness* são selecionados e enviados para o *crossover*.

Para o método de *crossover* com um ponto de corte realiza o cruzamento entres os pais (pai e mãe). A figura 13 apresenta os indivíduos antes do cruzamento e o indivíduo pai está marcado onde irá acontecer o ponto de corte.

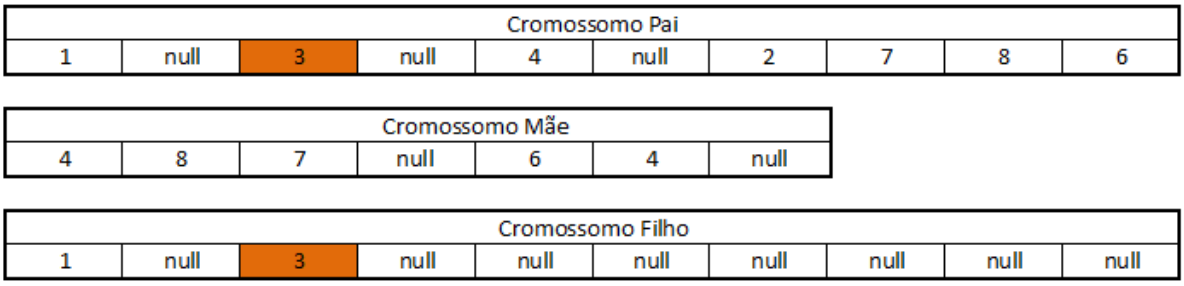
Figura 13 - Indivíduos antes do *crossover*



Fonte: Elaborada pelo autor

Após a criação o filho 1, ele irá recebe todos os genes contidos no pai que estão antes do valor do ponto de corte. Devemos lembrar que o filho é um indivíduo e que os mesmo já contem o numero de genes pré-definidos e os relacionamentos de obrigatoriedade entre disciplina e horário são nulos quando criados. Quando ocorre a troca genética de crossover, estamos falando do envio das obrigatoriedades das disciplinas em relação aos horários. Para que não ocorra a repetição das informações contidas no pai e na mãe, os genes já utilizados para compor os genes do filho 1 no ponto de corte são removidos do indivíduo mãe. A marca no cromossomo filho representa o ponto de corte, conforme apresentado na figura 14.

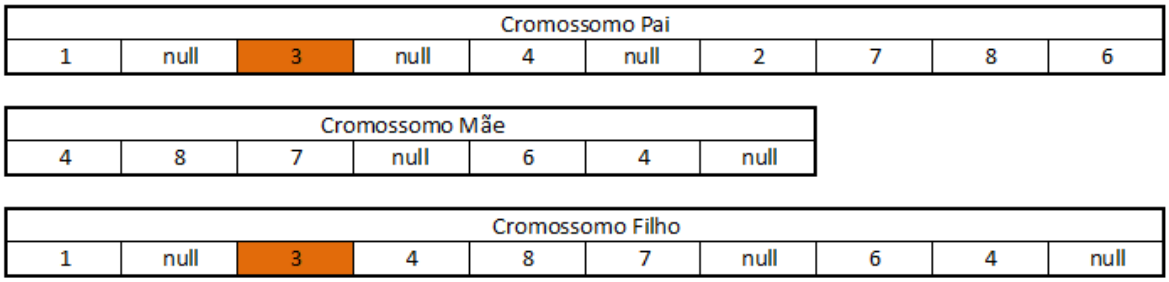
Figura 14 - Inserção do material genético do pai



Fonte: Elaborada pelo autor

Ao término da adição dos genes do ponto de corte o filho 1 recebe as informações genéticas que faltavam da mãe. A figura 15 representa como os indivíduos ficaram após a operação.

Figura 15 - Inserção do material genético da mãe



Fonte: Elaborada pelo autor

Este mesmo processo é realizado para a criação do filho 2, porém, ao criar este novo filho, devemos iniciar este mesmo processo descrito porem invertendo onde for pai, para mãe e vice versa as operações realizadas continuam as mesmas. Após a realização do *crossover* os filhos recebem uma nota de *fitness*.

4.2.4 População

A manipulação da população e de suas propriedades é feita através de parâmetros, enviados antes da execução do algoritmo. Estes parâmetros são elitismo, taxa de *crossover*, taxa de mutação, tamanho da população e número máximo de gerações.

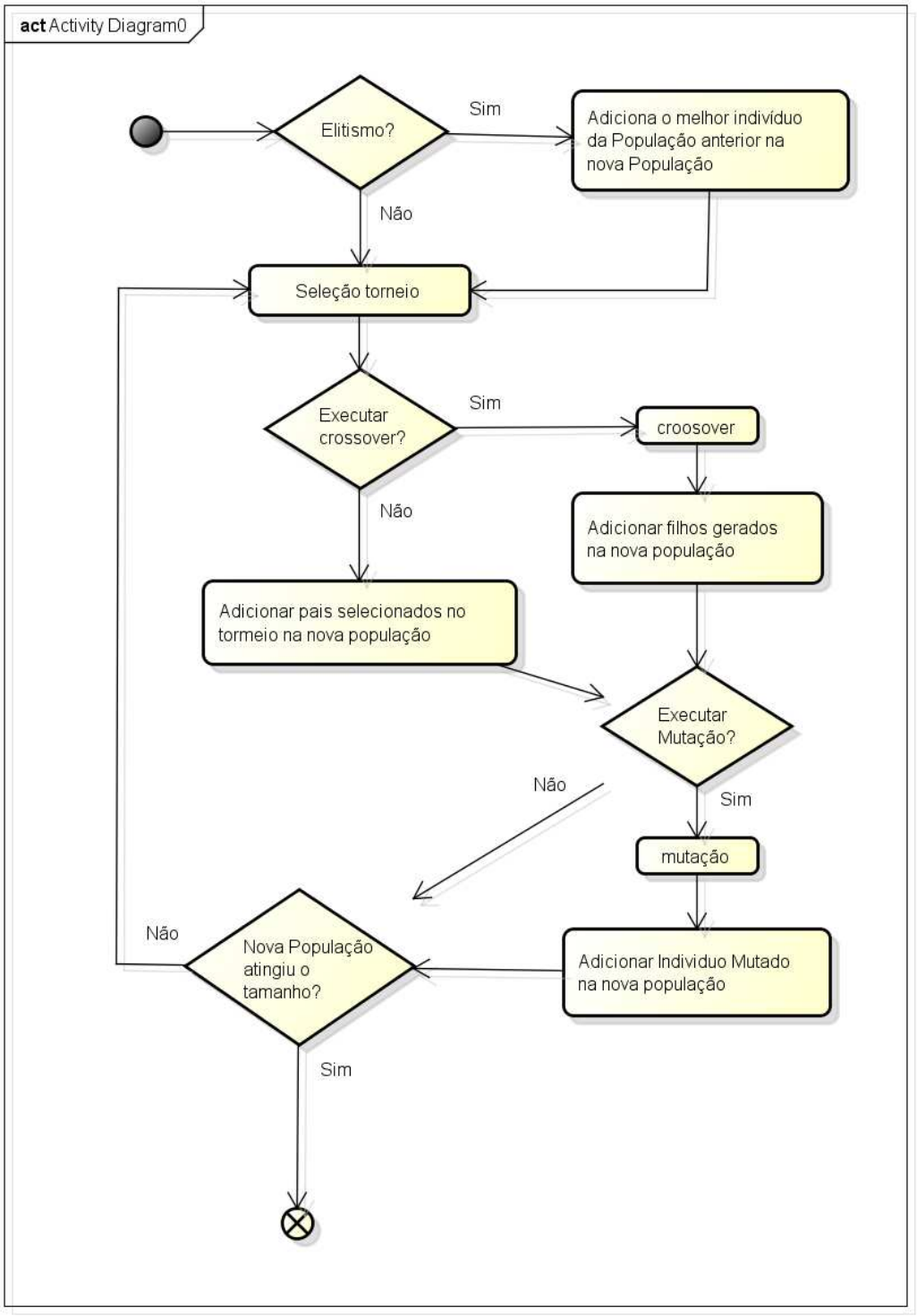
De acordo com os parâmetros passados é criada a população inicial. Para cada indivíduo criado é utilizado um método para inserir randomicamente todos os registros de “relacionamento de obrigatoriedade” entre disciplina e horário, em cada um dos genes que previamente foram criados como nulos. Estes indivíduos são criados até que a população tenha o tamanho da igual ao parâmetro tamanho da população.

Para cada geração, é criada uma nova população a partir da população criada na geração anterior. Se o operador genético elitismo estiver com o valor verdadeiro, iniciamos está nova população com o melhor indivíduo da população anterior. O melhor indivíduo de uma população é indicado pela maior pontuação de *fitness*.

Durante a criação da nova população podemos ter duas operações ocorrendo mutação e *crossover*. Para a execução destes operadores genéticos são utilizadas porcentagens enviadas pelos parâmetros do algoritmo. Para que estas operações genéticas aconteçam são utilizados valores randômicos para serem comparados com as taxas de mutação e *crossover*. Em cada interação da criação desta nova população são selecionados por torneio dois indivíduos que serão denominados como pais para serem utilizados na operação de *crossover*. Se a condição da taxa de execução for verdadeira, os pais serão descartados e os filhos serão gerados a partir dos genes dos pais através do operador de *crossover*. Logo em seguida serão adicionados na nova população. Em caso de falso os pais serão os indivíduos adicionados nesta nova população.

Para utilizar a operação de mutação também é utilizado um valor randômico, se a condição para realizar a mutação for verdadeira, um indivíduo da população anterior é selecionado e o mesmo sofrerá a mutação genética, o indivíduo antes da mutação genética é descartado e o novo indivíduo que sofreu a mutação é adicionado na nova população. O fluxo de uma nova população descrito pode ser visualizado na figura 16.

Figura 16 - Fluxo Nova População

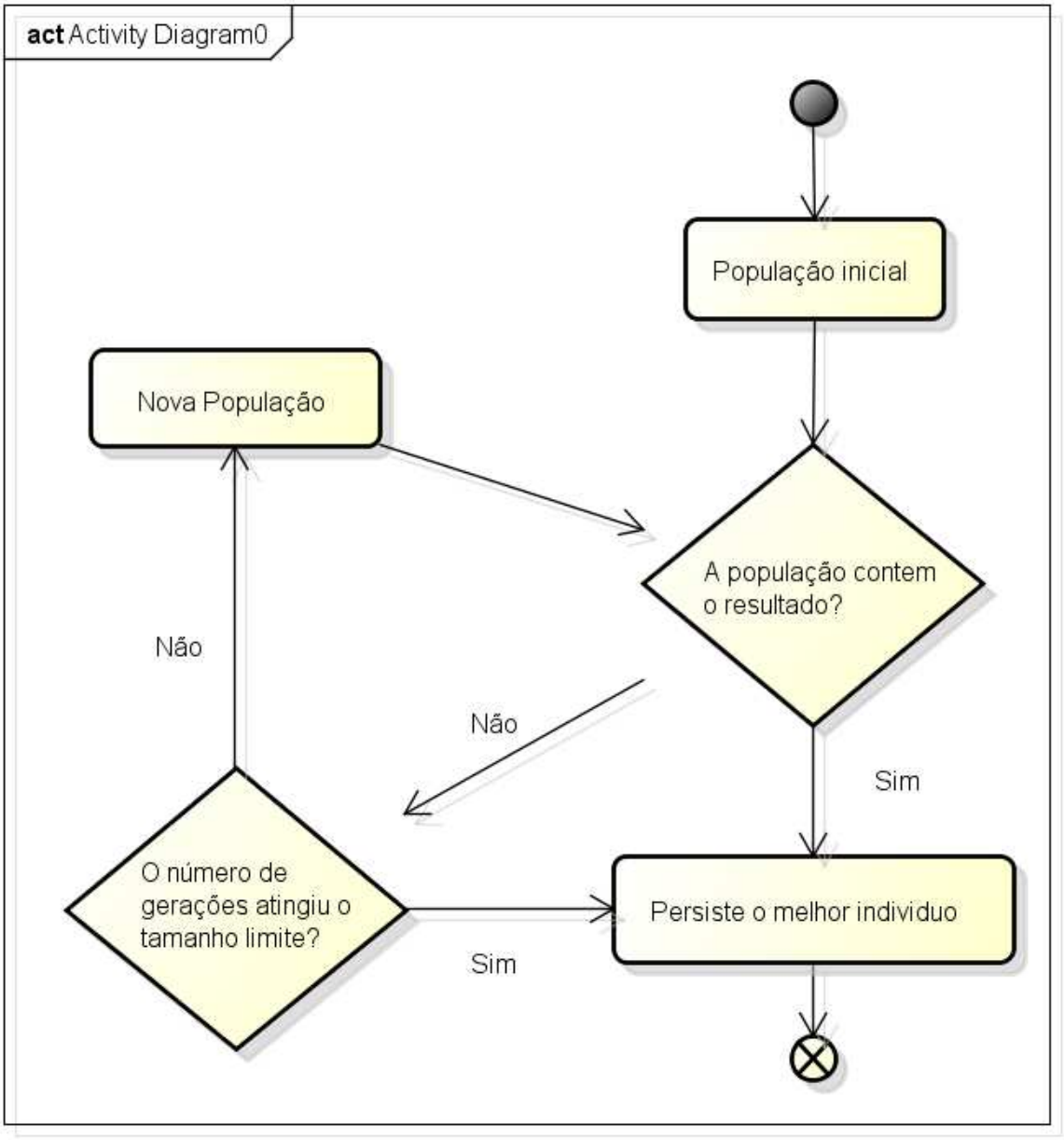


Fonte: Elaborada pelo autor

4.2.5 Fluxo do Algoritmo

O fluxo do algoritmo conforme a figura 17 é iniciada pela criação da população inicial. Para cada interação do algoritmo é verificado se a população contém o resultado e se o algoritmo não atingiu o numero de gerações pré-definidas. Se as duas condições forem falsas o algoritmo cria uma nova população.

Figura 17 - Fluxo do algoritmo



Fonte: Elaborada pelo autor

5 RESULTADOS OBTIDOS

Após a implementação do sistema foi realizada uma bateria de testes, com carga parcial dos dados, para que os valores dos parâmetros iniciais fossem calibrados para a carga completa. Com a definição dos melhores parâmetros a serem utilizados foi realizado um teste com a carga completa.

Os testes foram realizados em uma maquina com a seguinte configuração, *Intel (R) Core (TM) i5 3.40GHz* com *16Gb* de memória *RAM* sob o sistema operacional *Windows* 7 *64bit.*

5.1 Teste Inicial

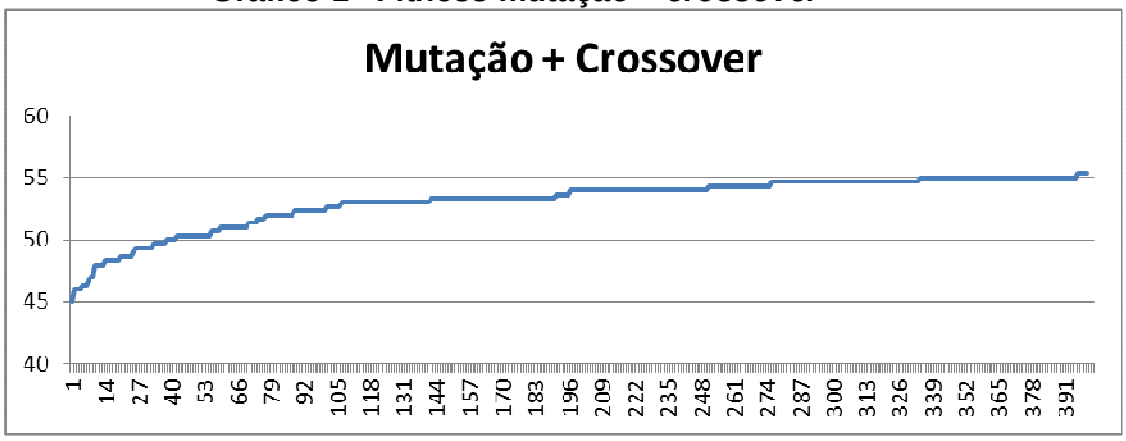
Os dados utilizados nesta primeira bateria têm o total de 10 salas e um curso com o total de 46 disciplinas e suas respectivas restrições de horário. Foram utilizados como parâmetros iniciais os dados representados na figura 18, uma população com 140 indivíduos, 60% de *crossover* 20% de mutação. Para descobrir qual será o desempenho do algoritmo antes de utilizar a carga completa, inicialmente o parâmetro de gerações foi ajustado para 400. O parâmetro elitismos está determinado como verdadeiro para todos os testes realizados. Estes parâmetros foram definidos por base na conclusão do trabalho de Ciscon, (2006).

Figura 18 - Parâmetros utilizados para o teste inicial

Fonte: Elaborada pelo autor

Ao realizar o primeiro teste, que utiliza os parâmetros iniciais definidos anteriormente, foi identificado que o algoritmo fica preso em planícies, isso quer dizer que o mesmo fica muito tempo sem ter uma evolução, ou seja, sem alterar o seu valor de *fitness*. Neste caso podemos notar no gráfico 1, que o algoritmo no começo tendo uma evolução rápida atingindo o valor de *fitness* 50, nas primeiras 50 gerações, porem na geração 100 até a 400 o algoritmo começa a perder seu poder de evolução caminhando devagar e atingindo o valor 55 em seu *fitness* em sua evolução total.

Gráfico 1 - *Fitness* mutação + *crossover*

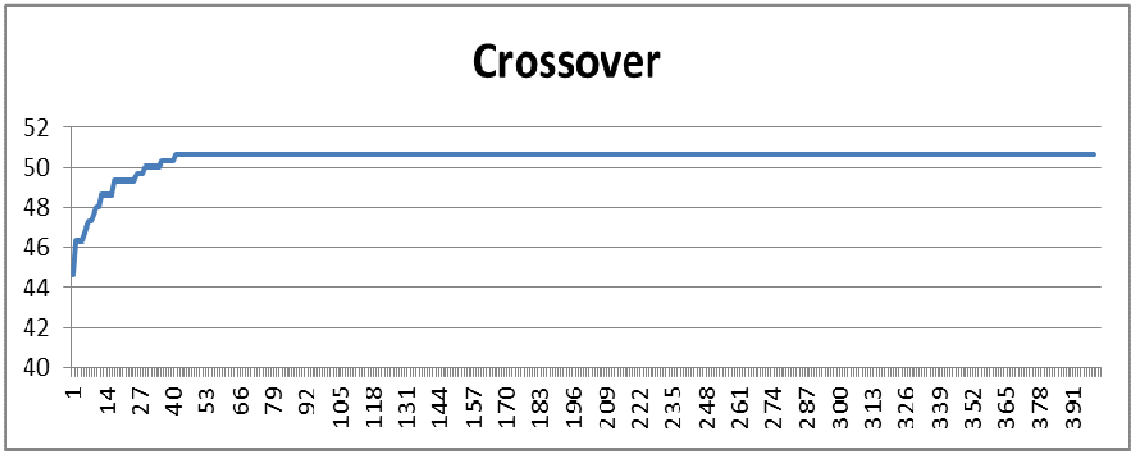


Fonte: Elaborado pelo autor

Tentando identificar as possíveis causas desta desaceleração e estagnação em planícies, foram realizados testes com alteração de alguns parâmetros para identificar o que estava acontecendo. No primeiro teste realizado o parâmetro de mutação não é utilizado. O gráfico 2 mostra o desempenho do algoritmo nesta situação.

Podemos observar que em determinado ponto o algoritmo evolui rapidamente atingindo o *fitness* 50 em 30 gerações, porém o mesmo entra em uma planície e não consegue mais prosseguir com sua evolução. Isso se deve ao fato de um super-indivíduo ter dominado a população passando a características dos seus genes para todos os indivíduos da população. Isto acontece, pois não ocorre a mutação que é responsável pela troca de genes de um indivíduo.

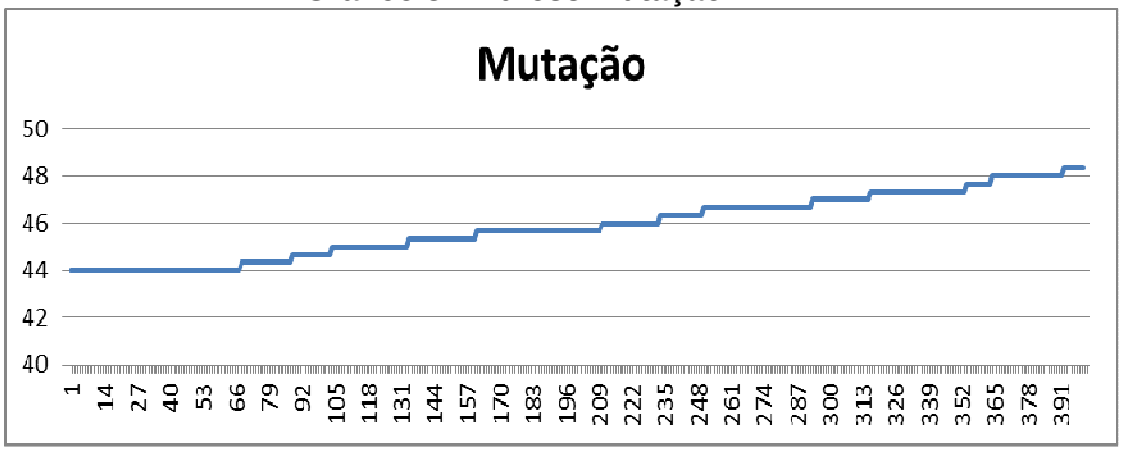
Gráfico 2 - *Fitness* Crossover



Fonte: Elaborado pelo autor

Logo em seguida foi realizado o teste em que o parâmetro de *crossover* não é utilizado. No gráfico 3 podemos observar que, a evolução através da mutação é bem lenta e o algoritmo encontra varias planícies durante a evolução, assim o processo para encontrar a solução se torna mais lento, devemos observar que com estas configurações o algoritmo não conseguiu atingir o *fitness* 50 em 400 gerações.

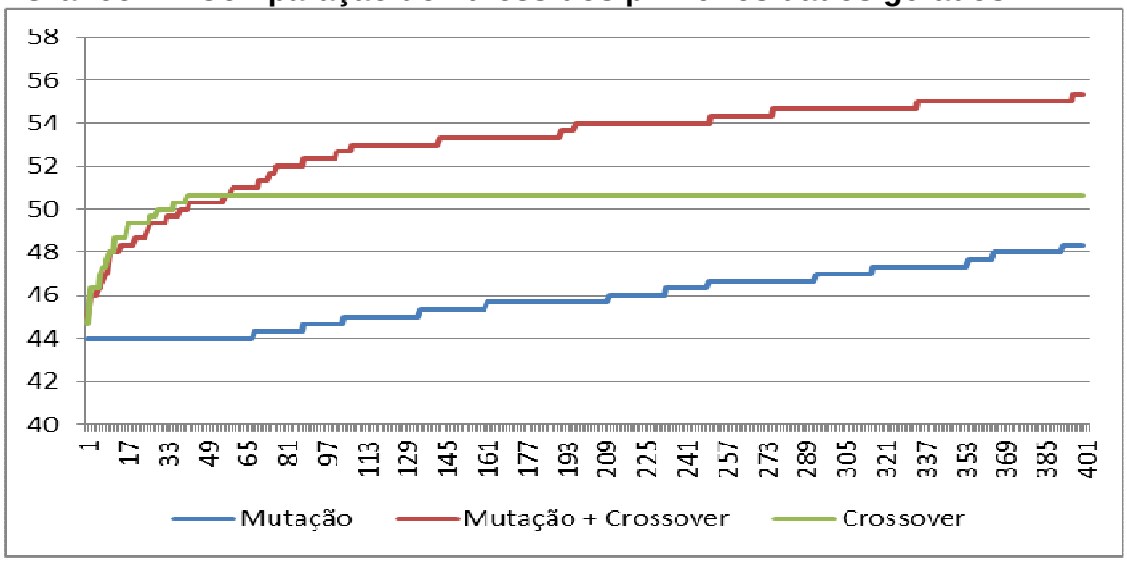
Gráfico 3 - *Fitness* mutação



Fonte: Elaborado pelo autor

Comparando os três gráficos gerados, representado pelo gráfico 4, podemos observar que o *crossover* é responsável por acelerar a evolução e a mutação faz com que o algoritmo mantenha a biodiversidade de indivíduos para que não ocorra a convergência em planícies.

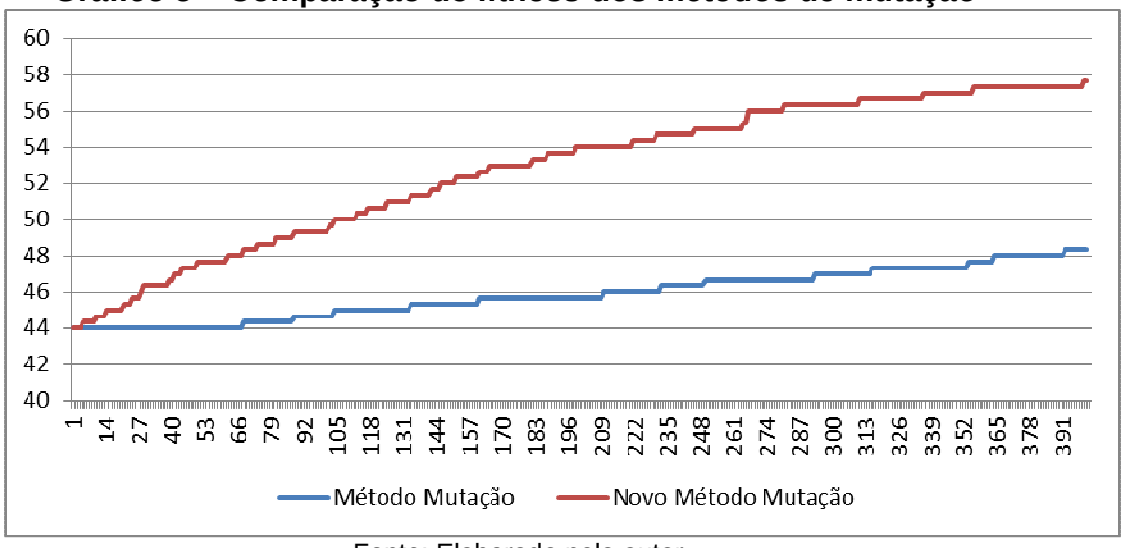
Gráfico 4 – Comparação do *fitness* dos primeiros dados gerados



Fonte: Elaborado pelo autor

Para acelerar a evolução do algoritmo, como solução o método de mutação foi alterado para que aconteça a melhoria genética. Com esta modificação, ao invés de apenas realizar uma troca randômica dos genes, ele procura fazer esta troca, enviando as características de um gene para um lugar mais adequado respeitando a restrições do problema. Novamente foi rodado um teste para verificar o desempenho da nova versão do método. O gráfico 5 representa a comparação entre a evolução da primeira versão do método com a nova versão que engloba a melhoria genética. Podemos observar uma grande melhoria de evolução por parte do operador de mutação o mesmo atinge o ponto de fitness 50 em 105 gerações.

Gráfico 5 – Comparação do *fitness* dos métodos de mutação



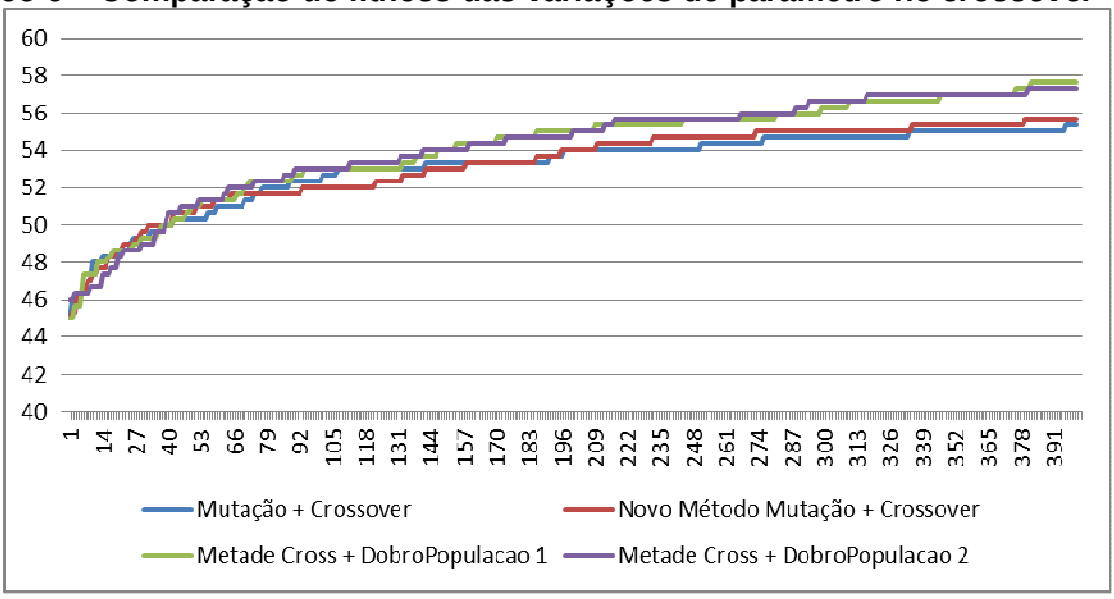
Fonte: Elaborado pelo autor

Para identificar o desempenho com o novo método de mutação novamente foi realizado um teste onde o parâmetro de *crossover* é adicionado e os outros parâmetros são mantidos como anteriormente. Porém foi observado que o resultado é bem próximo ao resultado que foi coletado com a utilização do método de mutação antigo onde o fitness. Por terem resultados bem parecidos, novamente foi detectada a criação de um super-indivíduo.

Tentando solucionar o problema de falta de biodiversidade, foi reduzido pela metade o parâmetro de *crossover* e aumentado para o dobro o número de indivíduos em uma população. Novamente com o resultado obtido após o teste foi identificada uma curva parecida com as geradas anteriormente, porém com uma pequena melhoria na evolução.

O mesmo acontece quando o parâmetro de *crossover* é novamente reduzido pela metade e o tamanho da população dobrado novamente a partir das modificações nos parâmetros realizadas anteriormente. A comparação dos resultados é apresentada no gráfico 6.

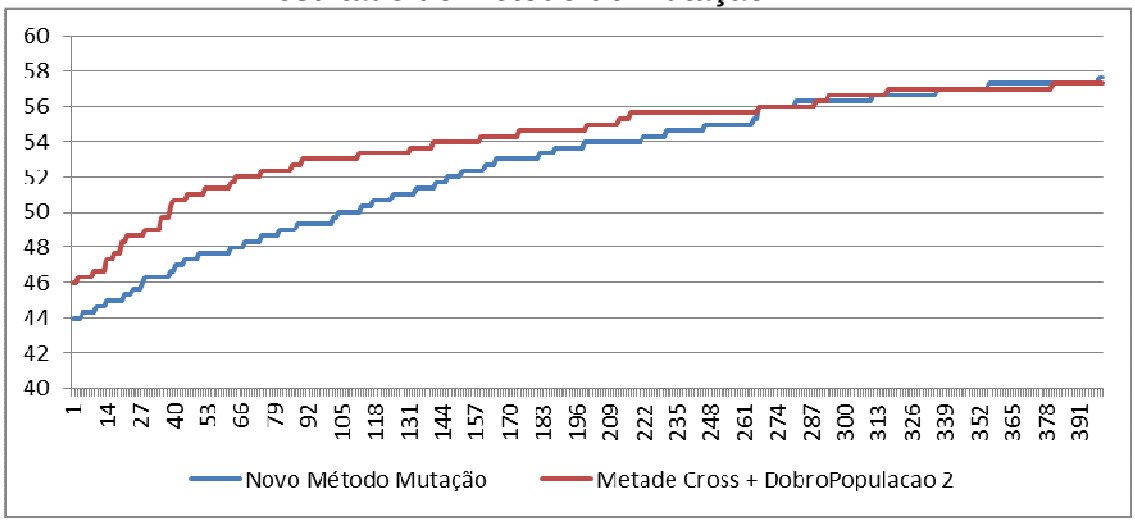
Gráfico 6 – Comparação do *fitness* das variações de parâmetro no *crossover*



Fonte: Elaborado pelo autor

Comparando o resultado da última utilização de mutação e *crossover* juntos, com o resultado da utilização do operador mutação sozinho, podemos identificar através da representação do gráfico 7 que os dois resultados têm chances de chegar ao resultado ótimo porem em diferentes gerações.

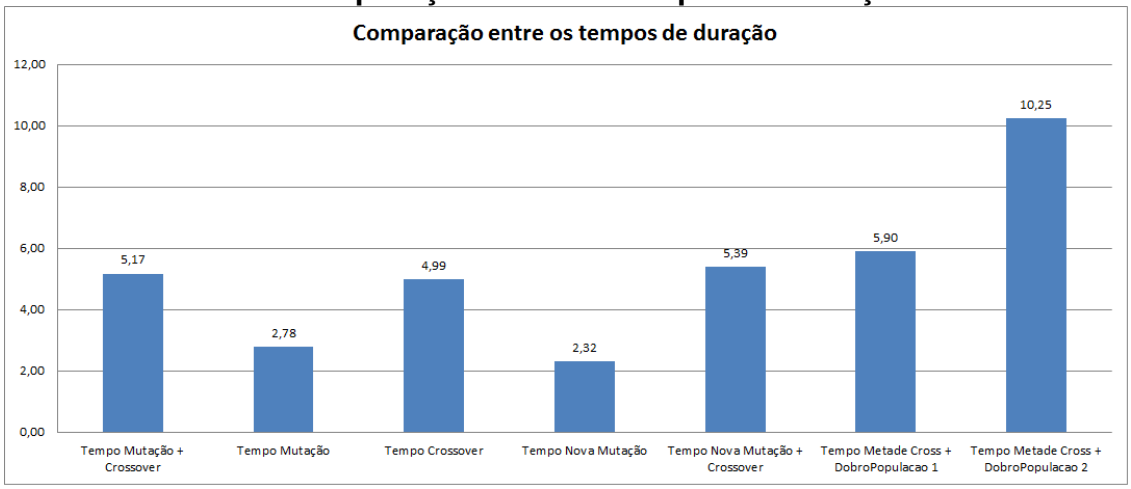
Gráfico 7 - Comparação ultimo resultado mutação e crossover juntos com resultado do método de mutação



Fonte: Elaborado pelo autor

O gráfico 8 representa uma comparação em horas, de todos os tempos de execução dos testes descritos acima. Após análise dos resultados do algoritmo e a comparação dos tempos de execução optou-se pela utilização apenas do método de mutação principalmente pelo tempo gasto para execução do mesmo em caso de comparação podemos observar que o novo método de mutação tem como desempenho 2h e o método que utiliza metade do *crossover* e o dobro da população e demora 10 horas para atingir o fitness 58 conforme mostrado no gráfico 7.

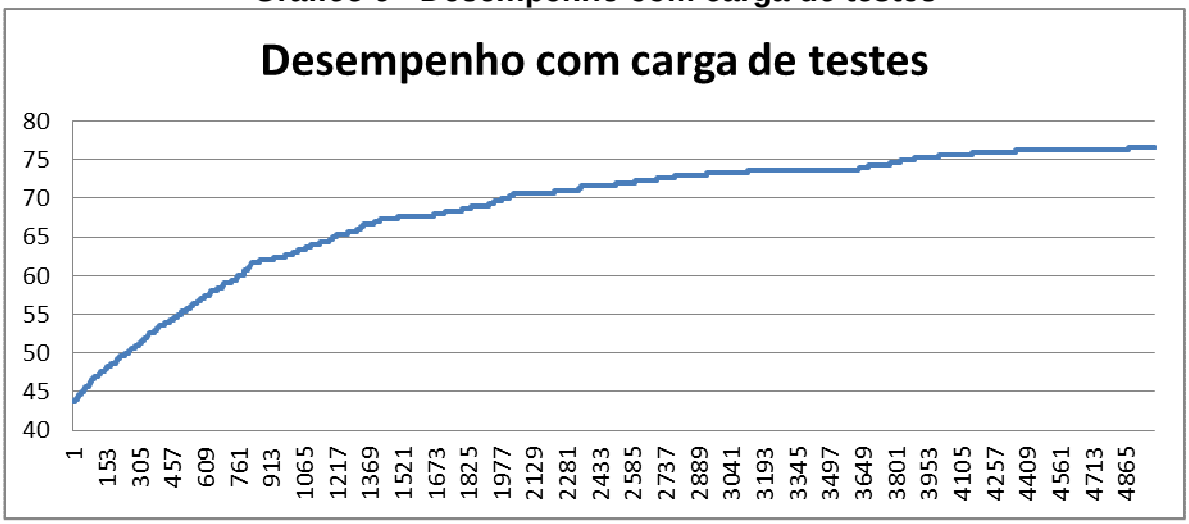
Gráfico 8 - Comparação entre os tempos de duração



Fonte: Elaborado pelo autor

Com o total de 30 horas e 5000 gerações o algoritmo encontra uma solução que atende a 70% das restrições pré-definidas. O gráfico 9 representa a curva de desempenho do algoritmo com a carga inicial dos testes.

Gráfico 9 - Desempenho com carga de testes



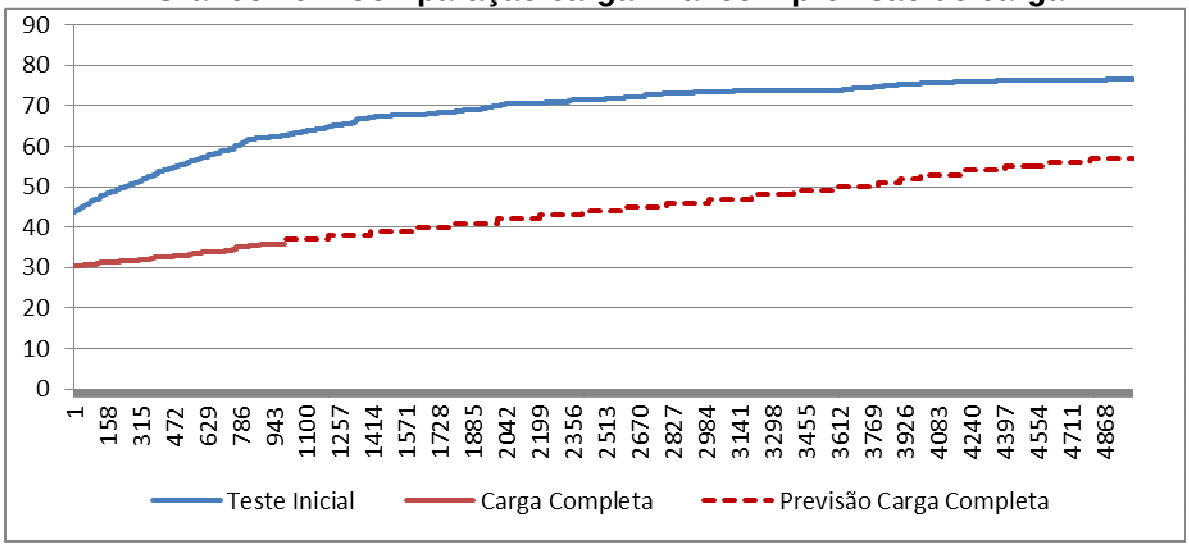
Fonte: Elaborado pelo autor

5.2 Teste Carga Completa

Ao iniciar os testes com a carga completa do sistema, o algoritmo tem como desempenho o valor de 7 min para cara geração. Levando em consideração os dados do teste com a base inicial, o algoritmo tem previsão de 90horas para finalizar as 5000 gerações, um valor muito alto.

O gráfico 10 mostra a curva da carga completa comparada com a carga de testes nas primeiras 1000 gerações. Podemos observar que o desempenho do algoritmo é bem menor e que levariam mais do que 5000 gerações para atingir o valor de 70% como anteriormente. O gráfico 10 também apresenta uma tendência para a solução na curva tracejada o que mostra que o algoritmo pretende atingir um fitness de 57 ao final das 5000 mil gerações, o que quer dizer a evolução de 27 pontos em seu *fitness*. Utilizando este valor a previsão do que é de 1% a cada 200 gerações, o algoritmo pretende encontrar uma solução ótima de 100% em 13000 gerações e o que gastaria um total de 63 horas o que levaria em torno de 3 dias para encontrar a solução tratando linearmente a previsão de carga.

Gráfico 10 – Comparação carga final com previsão de carga



Fonte: Elaborado pelo autor

6 CONSIDERAÇÕES FINAIS

Podemos observar que a implementação de um algoritmo genético não é uma tarefa trivial, pois o mesmo conta com varias técnicas para evitar que o algoritmo fique estacionado em planícies. Com isto é necessário que ocorram vários testes para poder identificar os problemas encontrados pelo algoritmo ao encontrar a solução ótima, uma vez que o tempo é um fator essencial ao se encontrar a solução.

O problema tratado que contem o total de 50 salas com 149 disciplinas é realizado hoje em dia manualmente e leva o tempo de duas semanas, 14 dias para ser realizado. Tendo que ser refeito caso aja alguma alteração. O sistema desenvolvido além de fazer um controle básico das informações através do CRUD teve uma previsão de 3 dias para encontrar uma solução que atenda as restrições definidas, além de ser um tempo menor ele não é feito manualmente o que já é um ganho para o gestor.

O algoritmo genético desenvolvido demonstra através das bases de teste que tem grandes chances de encontrar uma solução que atenda mais que 70% das restrições do problema. O número de gerações utilizadas se mostrou pequeno diante do tamanho do problema. Porém, o tempo gasto para encontrar a solução é muito grande. Isso mostra claramente que são necessárias novas avaliações do algoritmo para que o mesmo atinja valores mais significativos no tempo de execução a versão inicial do algoritmo gastou 30 h para gerar uma solução que atenda a 70% da base de testes utilizada e em torno de 3 dias pela previsão realizada com a carga total.

Podemos observar também que a utilização do operador genético de *crossover* faz com que no inicio das gerações o algoritmo tenha uma evolução muito rápida e com o tempo o conjunto de soluções no caso os indivíduos tenham uma convergência prematura em uma solução não ótima. Isto ocorre pelo fato de um super-indivíduo passar suas características para todos os indivíduos da população.

Contudo podemos observar que o algoritmo pode não achar uma solução que atenda a 100% das restrições necessárias pelo gestos, mas uma alocação inicial facilita o desgaste manual ao se fazer a primeira alocação que está passível de alterações devido as demandas envidas pelos colegiados.

6.1 Trabalhos Futuros

Aplicação de técnicas para evitar que o algoritmo tenha sua convergência prematura, com isto o algoritmo levara menos tempo para encontra uma solução que atenda as restrições do problema de forma mais rápida. Como exemplo podemos notar que a população inicial começa com um valor de fitness muito baixo menos da metade, com isto podemos adotar técnicas para que a população inicial tenha já encontre uma solução com 80% ou mais para que o algoritmo gaste menos tempo com evoluções desnecessárias.

Realizar novos testes, porem com parâmetros diferentes para identificar o desempenho do algoritmo. Podemos testar o algoritmo com alterando os valores de mutação que neste trabalho não foi abordado.

Utilização de outros algoritmos como Busca Tabu, Recozimento Simulado para tentar solucionar o mesmo problema, com a mesma carga e restrições. Com os resultados avaliar o desempenho destes algoritmos, dificuldades de implementação e possíveis problemas encontrados ao encontrar o ótimo global.

REFERÊNCIAS

AARTS, E.; KORST, J. **Simulated annealing and boltzmann machines.** New York, NY; John Wiley and Sons Inc., 1988.

ARMENTANO, V. A.; BRANCHINI, R. M. **Uma introdução à busca tabu.** 2013.

CAELUM. Apostila do curso FJ-11 - **Java e Orientação a Objetos**. 2013. Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos>. Acesso em: 29 set. 2013.

CARTER, M. W.; TOVEY, C. A. **When is the classroom assignment problem hard? Operations Research**, INFORMS, v. 40, n. 1-Supplement-1, p. S28–S39, 1992.

CISCON, L. A. **O problema de geração de horários: Um foco na eliminação de janelas e aulas isoladas.** 2006.

DARWIN, C. **On the origin of species by means of natural selection.** 1859.

EVANS, J. R.; MINIEKA, E. **Optimization algorithms for networks and graphs.** [S.l.]: CRC Press, 1992.

EVEN, S.; ITAI, A.; SHAMIR, A. **On the complexity of time table and multi-commodity flowproblems.** In: IEEE. Foundations of Computer Science, 1975., 16th Annual Symposium on. [S.l.], 1975. p. 184–193.

GLOVER, F. **Future paths for integer programming and links to artificial intelligence. Computers & Operations Research**, Elsevier, v. 13, n. 5, p. 533–549, 1986.

GÓES, A. R. T. **Otimização na Distribuição da Carga Horária de Professores: método exato, método heurístico, método misto e interface.** Tese (Doutorado) - Dissertação de Mestrado, UFPR, 2005.

GOLBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear.** Rio de Janeiro: Campus, 2000.

GOLDBERG, D. **Genetic algorithms in optimization, search and machine learning. Addison** Wesley, New York. Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing.

Springer. Jacq J, Roux C (1995) **Registration of non-segmented images using a genetic algorithm.** Lecture notes in computer science, v. 905, p. 205–211, 1989.

HAMAWAKI, C. D. L. **Geração automática de grade horária usando algoritmos genéticos: o caso da faculdade de engenharia elétrica da ufu.** 2011.
HOLLAND, J. H. **Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.** [S.l.]: U Michigan Press, 1975.

JONG, K. A. D. **An analysis of the behavior of a class of genetic adaptative systems. Tese (Doutorado)** - Dissertation Abstracts International, 36(10),5140B. (University Micro lms No. 76-9381), 1975.

KIRKPATRICK, S.; JR., D. G.; VECCHI, M. P**. Optimization by simmulated annealing.** science, Washington, v. 220, n. 4598, p. 671–680, 1983.

LACERDA, E. G. de; CARVALHO, A. de. **Introdução aos algoritmos genéticos. Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais**, v. 1, p. 99–148, 1999.

LOBO, E. L. M. **Uma solução do problema de horário escolar via algoritmo genético paralelo.** Centro Federal de Educação Tecnológica de Minas Gerais, 2005.

LUCAS, D. C. **Algoritmos genéticos: um estudo de seus conceitos fundamentais e aplicação no problema de grade horária.** 2000.

LUVEZUTE, R. M.; KRIPKA, K. M. **Simulated annealing aplicado ao problema de alocação de salas com deslocamentos mínimos.** 2013.

MARINHO, **E. Heurísticas busca tabu para o problema de programação de tripulações de ônibus urbano.** Tese (Doutorado) — Master's Thesis, Universidade Federal Fluminense, 2005.

MICHALEWICZ, Z.; SCHOENAUER, M. **Evolutionary algorithms for constrained parameter optimization problems.** Evolutionary computation, MIT Press, v. 4, n. 1, p. 1–32, 1996.

MITCHELL, M. **An introduction to genetic algorithms (complex adaptive systems).** A Bradford Book, 1998.

NASCIMENTO, A. S.; SILVA, R. M. S.; **ALVARENGA, G. B. Uma aplicação de simulated annealing para o problema de alocação de salas.** INFOCOMP Journal of Computer Science, v. 4, n. 3, p. 59–66, 2005.

NORONHA, T. **Uma abordagem sobre estratégias metaheurísticas.** 2000

OLIVEIRA, A. C. de. **Uso do algoritmo genético e recozimento simulado para o problema de alocação de salas.** Monografia (Conclusão do curso) - Departamento de Ciência da Computação, Universidade Federal de Lavras, 2006.

OLIVEIRA, H. **Algoritmo evolutivo no tratamento do problema de roteamento de veículos com janela de tempo.** Monografia (Conclusão do curso) - Departamento de Ciência da Computação, Universidade Federal de Lavras. 2005.
PINHEIRO, P.; OLIVEIRA, J. A. **Um ambiente de apoio a construção de horário escolar na web: modelagem, implementação e aplicação nas escolas de ensino médio. XXXIII Simpósio Brasileiro de Pesquisa Operacional"**. Campos do Jordão, SP, 2001.

PLAY! **The High Velocity Web Framework For Java and Scala.** 2013. Disponível em: <http://www.playframework.com/>. Acesso em: 29 set. 2013.

POSTGRESQL. **Sobre o PostgreSQL.** 2013. Disponível em: <http://www.postgresql.org.br/sobre>. Acesso em: 29 set. 2013.

RAO, S. S. **Optimization : theory and applications.** New Delhi: Wiley Eastern, 1984. ISBN 0-85226-780-0. Disponível em: <http://opac.inria.fr/record=b1092847>. Acesso em: 25 set. 2013.

RAUPP, M. **Introdução à otimização linear.** LNCC, Rio de Janeiro. Notas de Aulas, Curso de Verão LNCC, 2003.

SCHAERF, A. **A survey of automated timetabling. Artificial intelligence review, Springer.** v. 13, n. 2, p. 87–127, 1999.

SILVA, A. **Estudo e implementação, mediante recozimento simulado, do problema de alocação de salas.** Monografia (Conclusão do curso) - Departamento de Ciência da Computação, Universidade Federal de Lavras, 2005.

SOUZA, M. J. F. **Programação de horários em escolas: uma aproximação por metaheurísticas.** Rio de Janeiro, 2000.

SOUZA, M. J. F.; MARTINS, A. X.; ARA ´ UJO, C. R. d. **Experiências com simulated annealing e busca tabu na resolução do problema de alocação de salas.** 2002.

STEIGLITZ, K.; PAPADIMITRIOU, C. H. **Combinatorial optimization: Algorithms and complexity. Prentice Hall, New Jersey., UV Vazirani (1984). On two geometric problems related to the travelling salesman problem.** J. Algorithms, v. 5, p. 231–246, 1982.

SUBRAMANIAN, A. et al. **Aplicação da metaheurística busca tabu na resolução do problema de alocação de salas do centro de tecnologia da ufpb.** Anais do XXVI Encontro Nacional de Engenharia de Produção, p. 1, 2006.

TIM ´ OTEO, G. T. S. **Desenvolvimento de um Algoritmo Genético para a Resolução do Timetabling**. 2005.

W3C. **HTML 4.01 Specification.** 2013 a. Disponível em: <http://www.w3.org/TR/html4>. Acesso em: 29 set. 2013.

W3C. **Cascading Style Sheets.** 2013 b. Disponível em: <http://www.w3.org/Style/CSS>. Acesso em: 29 set. 2013.

W3SCHOOLS. **JavaScript Tutorial.** 2013. Disponível em: <http://www.w3schools.com/js/>. Acesso em: 29 set. 2013.

APÊNDICE A - QUESTIONÁRIO DE ENTREVISTA

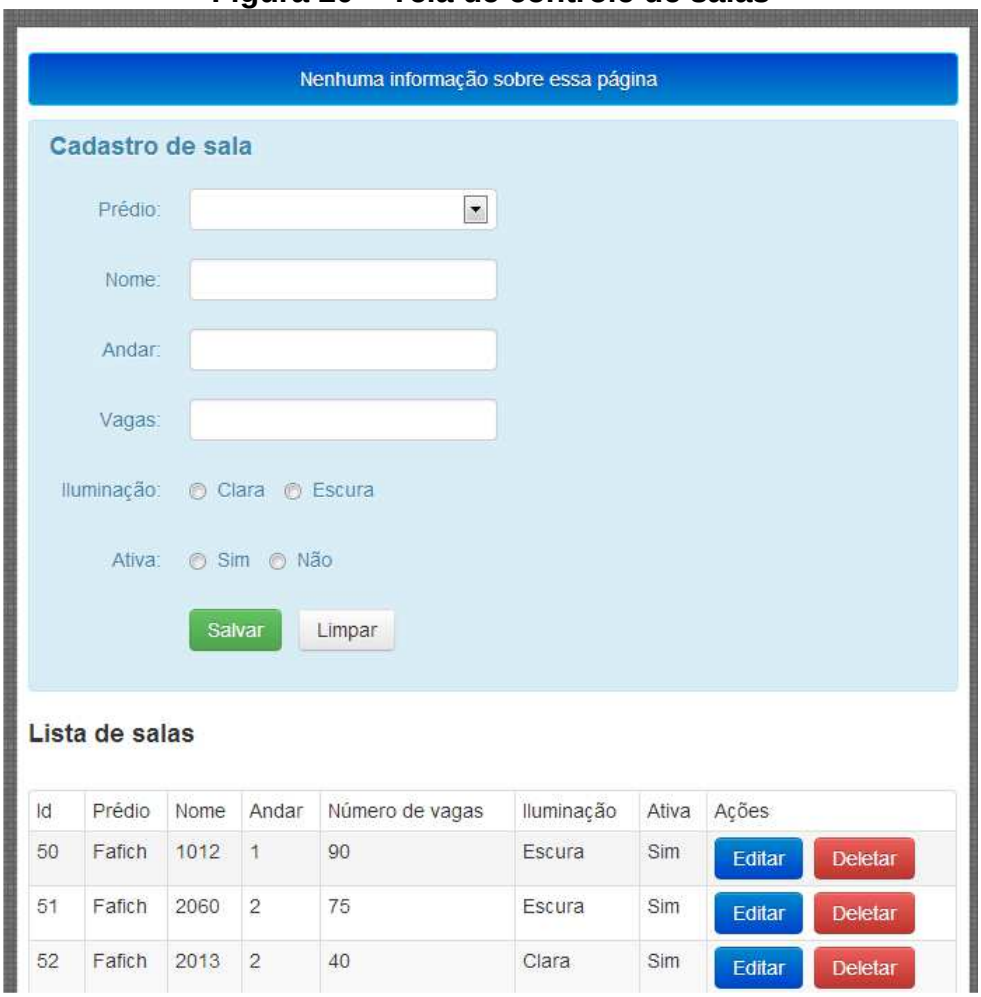
1) Caracterizar o problema de alocação
2) Quanto tempo em média demora a confecção manual da alocação?
3) Quais são os problemas mais comuns na alocação de salas?
4) Depois de realizada/divulgada a primeira versão da alocação ela sofre alterações? Por quê?

APÊNDICE B – PRINTS DAS TELAS SISTEMA

Figura 19 – Tela de controle de prédios

Fonte: Elaborado pelo autor

Figura 20 – Tela de controle de salas

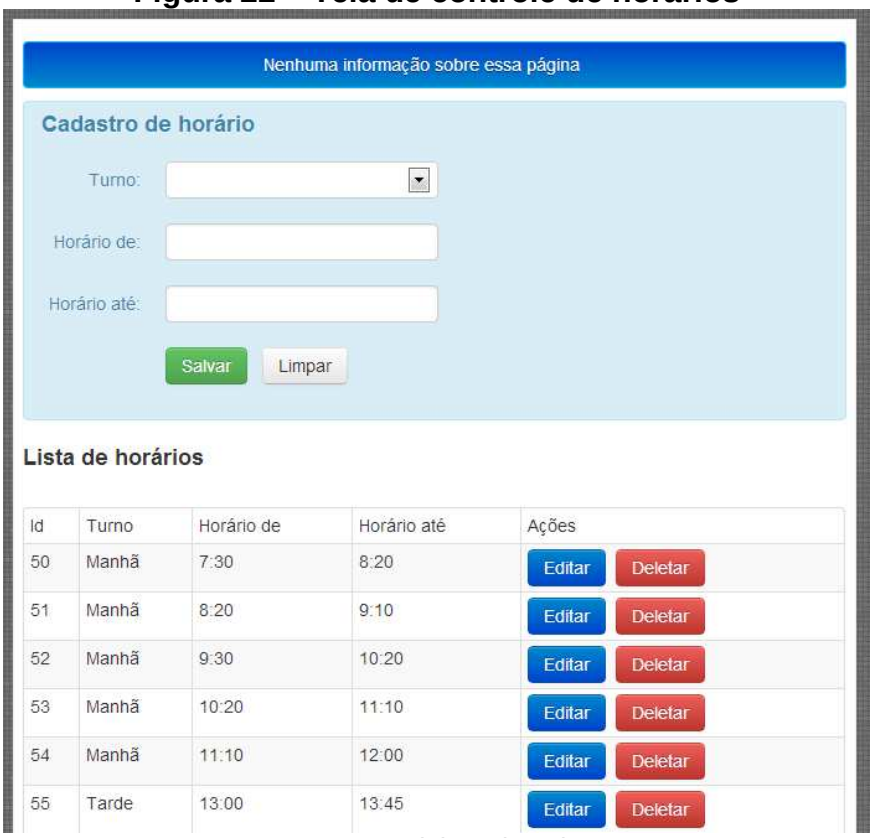


Fonte: Elaborado pelo autor

Figura 21 – Tela de controle de turnos

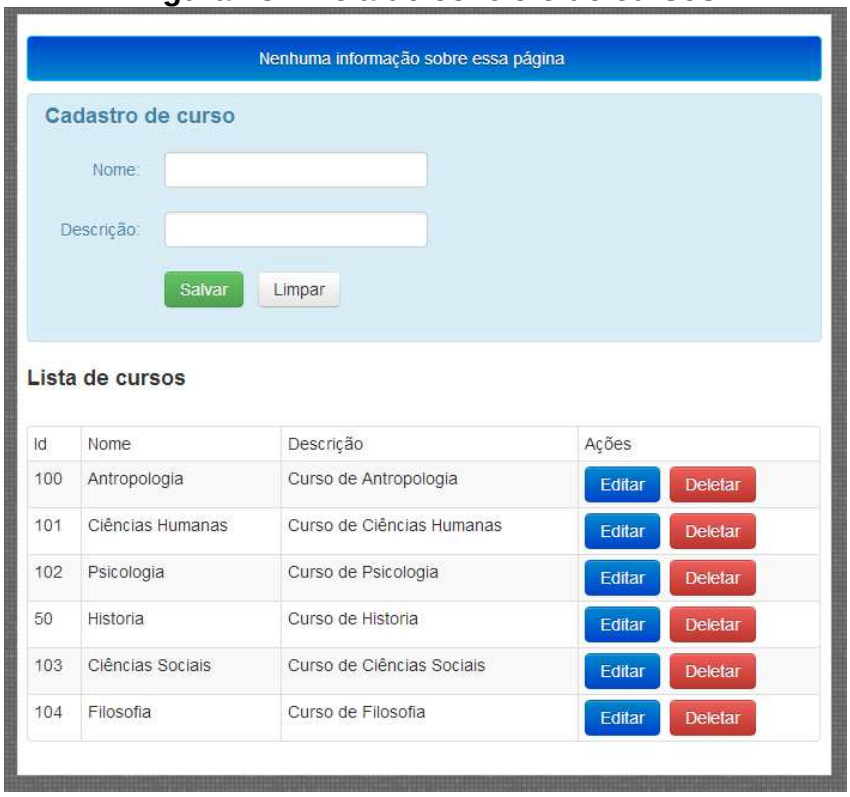
Fonte: Elaborado pelo autor

Figura 22 – Tela de controle de horários



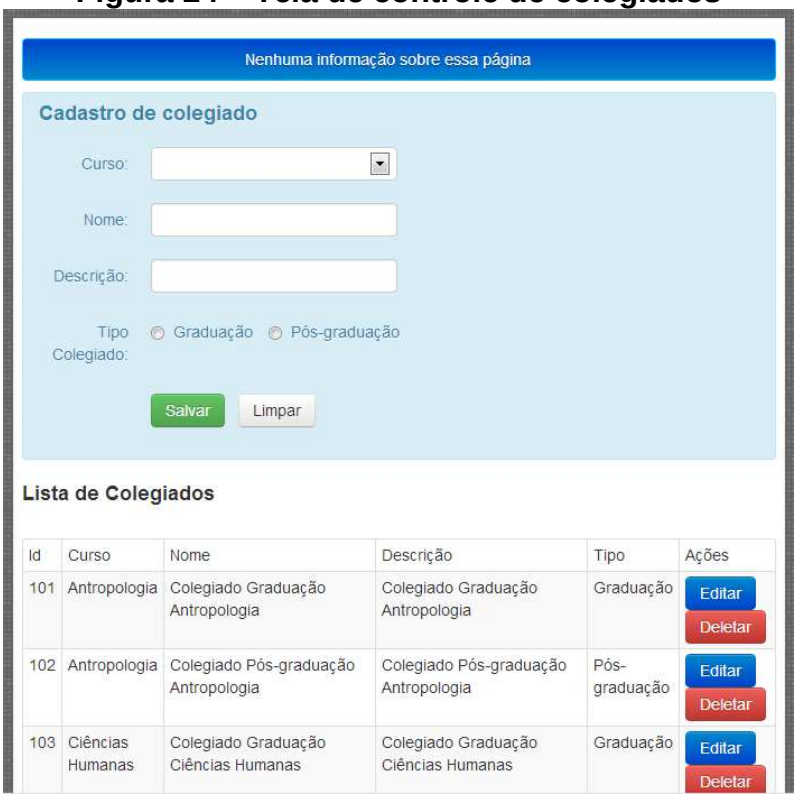
Fonte: Elaborado pelo autor

Figura 23 – Tela de controle de cursos



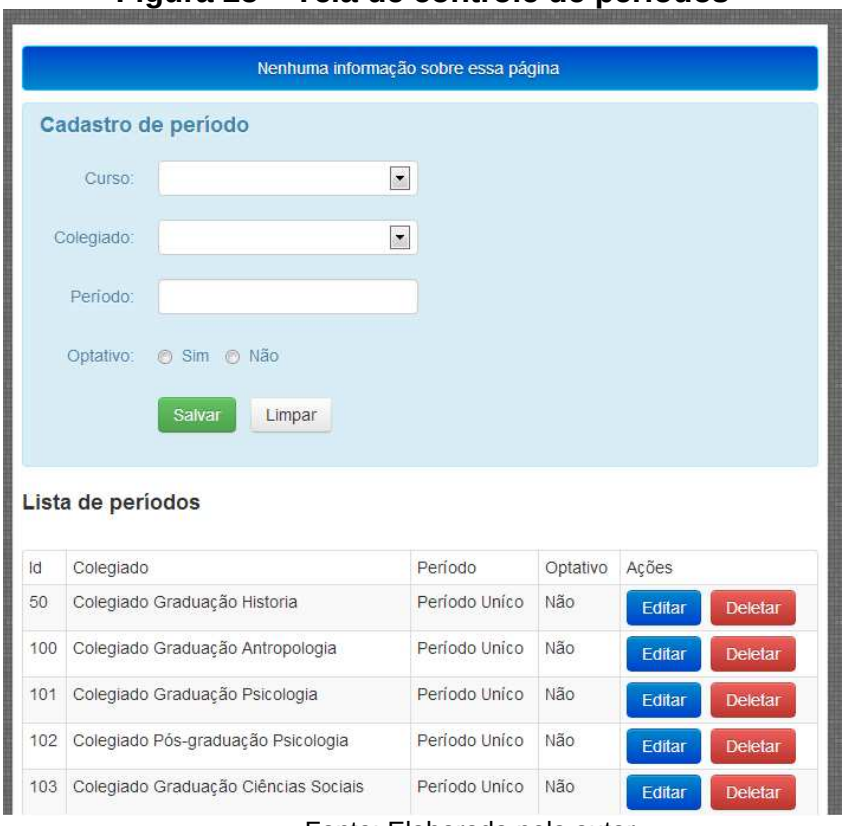
Fonte: Elaborado pelo autor

Figura 24 – Tela de controle de colegiados



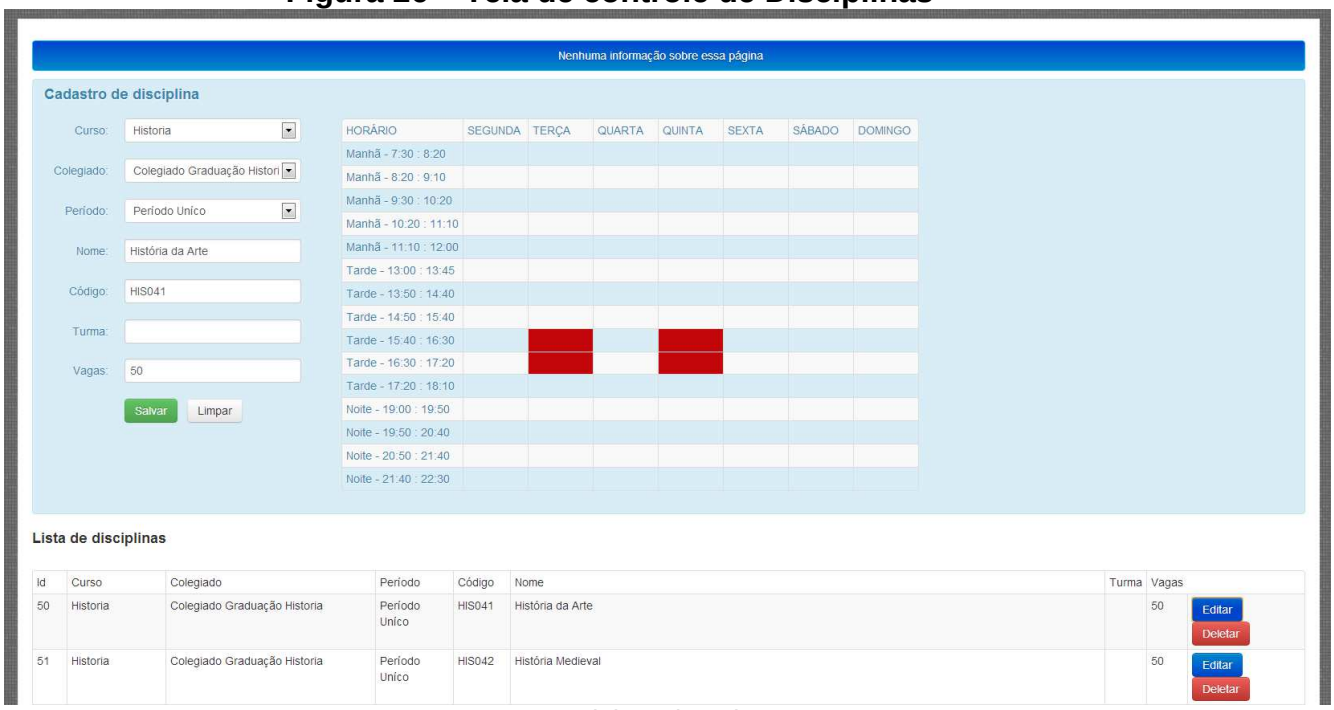
Fonte: Elaborado pelo autor

Figura 25 – Tela de controle de períodos



Fonte: Elaborado pelo autor

Figura 26 – Tela de controle de Disciplinas

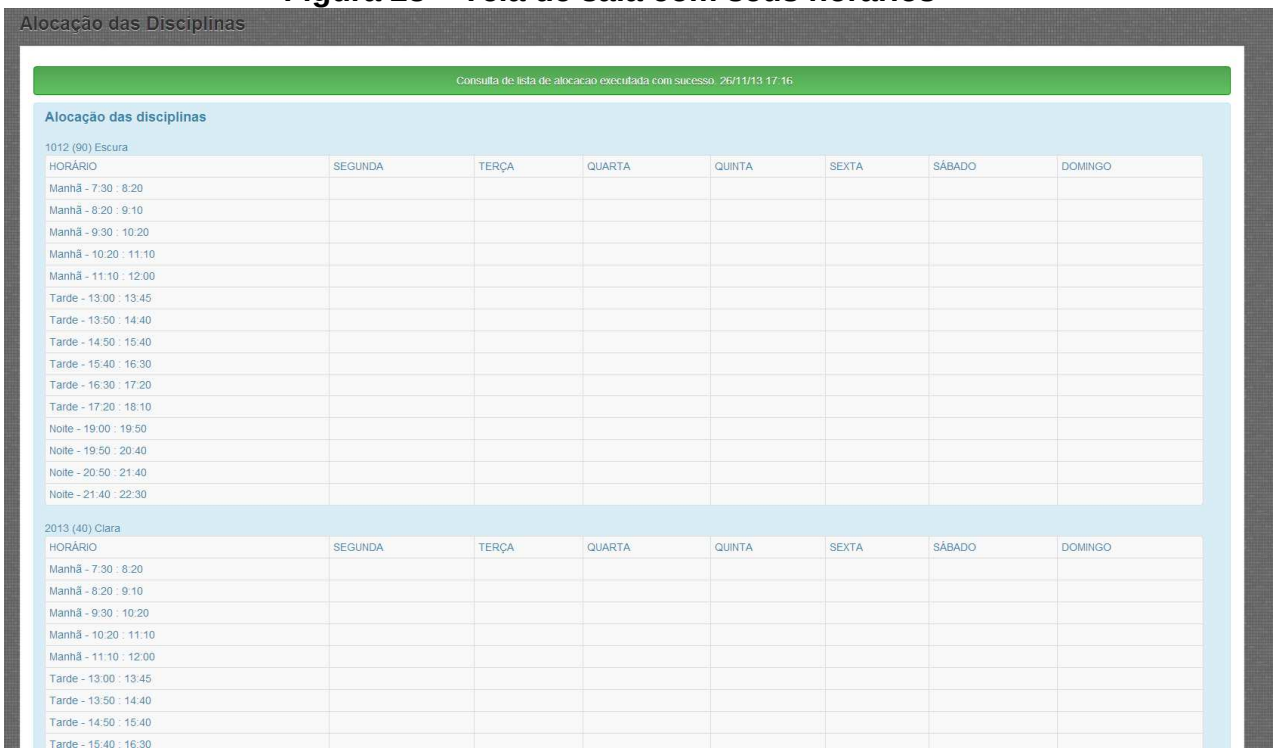


Fonte: Elaborado pelo autor

Figura 27 – Tela de configuração dos parâmetros do algoritmo genético

Fonte: Elaborado pelo autor

Figura 28 – Tela de sala com seus horários



Fonte: Elaborado pelo autor

ANEXO A – ALGORITMO BUSCA TABU

procedimento $BT(f(.), N(.), A(.), |V|, f_{min}, |T|, BTmax, s)$
1 $s^{\star} \leftarrow s$; {Melhor solução obtida até então}
2 $Iter \leftarrow 0$; {Contador do número de iterações}
3 $MelhorIter \leftarrow 0$; {Iteração mais recente que forneceu $s^{\star}$}
4 $T \leftarrow \emptyset$; {Lista Tabu}
5 Inicialize a função de aspiração A;
6 enquanto $(f(s) > f_{min}$ e $Iter - MelhorIter < BTmax)$ faça
7 $Iter \leftarrow Iter + 1$;
8 Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subset N(s)$ tal que
o movimento m não seja tabu $(m \notin T)$ ou
s' atenda a condição de aspiração $(f(s') < A(f(s)))$;
9 $T \leftarrow T -$ {movimento mais antigo} + {movimento que gerou s'};
10 Atualize a função de aspiração A;
11 $s \leftarrow s'$;
12 se $(f(s) < f(s^{\star}))$ então
13 $s^{\star} \leftarrow s$;
14 $MelhorIter \leftarrow Iter$;
15 fim-se;
16 fim-enquanto;
17 $s \leftarrow s^{\star}$;
18 Retorne s;
fim BT;

ANEXO B – ALGORITMO RECOZIMENTO SIMULADO

procedimento $SA(f(.), N(.), \alpha, SAmax, T_0, s)$
1 $s^{\star} \leftarrow s$; {Melhor solução obtida até então}
2 $IterT \leftarrow 0$; {Número de iterações na temperatura T}
3 $T \leftarrow T_0$; {Temperatura corrente}
4 enquanto $(T > 0)$ faça
5 enquanto $(IterT < SAmax)$ faça
6 $IterT \leftarrow IterT + 1$;
7 Gere um vizinho qualquer $s' \in N(s)$;
8 $\Delta = f(s') - f(s)$;
9 se $(\Delta < 0)$
10 então
11 $s \leftarrow s'$;
12 se $(f(s') < f(s^{\star}))$ então $s^{\star} \leftarrow s'$;
13 senão
14 Tome $x \in [0, 1]$;
15 se $(x < e^{-\Delta/T})$ então $s \leftarrow s'$;
16 fim-se;
17 fim-enquanto;
18 $T \leftarrow \alpha \times T$;
19 $IterT \leftarrow 0$;
20 fim-enquanto;
21 $s \leftarrow s^{\star}$;
22 Retorne s;
fim SA;