

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b>	<b>3</b>
1.1	Contextualização	3
1.2	Objetivos	3
1.2.1	Objetivos Gerais	4
1.2.2	Objetivos Específicos	4
1.3	Justificativa	4
1.4	Organização do Trabalho	4
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>6</b>
2.1	Problemas de Otimização	6
2.2	Heurística	7
2.2.1	Busca Tabu	8
2.2.2	Recozimento Simulado	9
2.2.3	Algoritmos genéticos	10
2.3	TimeTabling	13
2.4	Trabalhos Relacionados	14
<b>3</b>	<b>METODOLOGIA</b>	<b>16</b>
3.1	Ferramentas Utilizadas	17
3.1.1	Sistema de Gerenciamento de Banco de Dados	18
3.1.2	Ferramentas Back-end	18
3.1.3	Ferramentas Front-end	19
3.1.4	IDE	20
<b>4</b>	<b>SISTEMA DESENVOLVIDO</b>	<b>21</b>
4.1	Modelagem	21
4.1.1	Diagramas de caso de uso	21
4.1.2	Diagrama de Entidade Relacionamento	22
4.1.3	Diagramas de classe	23
4.2	Algoritmo Genético	24
4.2.1	Indivíduo	25
4.2.2	População	26
4.2.3	Operadores genéticos	28
4.2.4	Definição da função objetivo	30
4.2.5	Fluxo do algoritmo	30
<b>5</b>	<b>RESULTADOS OBTIDOS</b>	<b>32</b>
<b>6</b>	<b>CONSIDERAÇÕES FINAIS</b>	<b>33</b>
6.1	Trabalhos futuros	33

<b>REFERÊNCIAS . . . . .</b>	<b>34</b>
------------------------------	-----------

# 1 INTRODUÇÃO

## 1.1 Contextualização

Instituições de ensino universitárias se deparam todo início de semestre letivo com o problema de alocação de salas, este problema pode ser definido como *Classroom Assignment* que é uma instância do *course timetabling*, problemas desta instância são problemas de otimização combinatoria. Problemas de otimização combinatoria tem 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 em alguns casos pode demandar semanas de trabalho da pessoa responsável.

Em suma o trabalho consiste na distribuição das disciplinas pertencentes aos cursos de graduação e pós-graduação apresentados por alguns colegiados da Universidade Federal de Minas Gerais (UFMG), em salas disponibilizadas pelo prédio da Faculdade de Filosofia e Ciências Humanas (FAFICH). Para a distribuição destas disciplinas nas salas, foi criado um sistema que utiliza conceitos de algoritmo genético para resolver o problema PAS. Esta demanda de alocação acontece todo início de semestre e é executada assim que todos os colegiados tenham enviado suas solicitações necessárias para aquele semestre.

Uma vez que os termos da biologia utilizados foram ligados com o problema, o sistema gera uma alocação com grandes chances de atender as necessidades da instituição.

## 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. Com este trabalho objetiva-se:

### *1.2.1 Objetivos Gerais*

O objetivo deste trabalho é utilizar os conceitos de algoritmo genético para a resolução dos problemas denominados PAS através do desenvolvimento de um sistema que atenda todas as necessidades da instituição e facilite o gerenciamento das informações da instituição como salas, disciplinas e demais informações.

### *1.2.2 Objetivos Específicos*

- Desenvolvimento do sistema.
- Implementação de um algoritmo que proporcione uma solução de qualidade.
- Otimizar o tempo do gestor.
- Eficiência na geração dos relatórios.

## **1.3 Justificativa**

A solução de problemas de PAS através de meta-heurísticas se trata de uma área ainda não consolidada, por mais que existam trabalhos relacionados ao tema espera-se que as conclusões realizadas neste trabalho agreguem valor algum valor para os trabalhos futuros.

## **1.4 Organização do Trabalho**

Este trabalho está definido da seguinte forma, foi dividido em seis capítulos, sendo este capítulo 1 e mais cinco outros.

O capítulo 2 apresenta o referencial teórico do trabalho, descrevendo os conceitos utilizados para o desenvolvimento do projeto proposto: conceitos de TimeTable e Heurísticas.

No capítulo 3 é apresentada a metodologia do sistema e as tecnologias adotadas para desenvolvimento da solução.

No capítulo 4 iremos descrever e citar detalhadamente as características e propostas de

desenvolvimento do sistema desenvolvido, proposto para este trabalho.

A conclusão deste trabalho e considerações finais são mostrados nos capítulos 5 e 6.

## 2 REFERENCIAL TEÓRICO

### 2.1 Problemas de Otimização

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

De acordo com (STEIGLITZ; PAPADIMITRIOU, 1982) a constituição de um problema de otimização se deve aos termos vizinhança, ótimo local e ótimo global. O termo vizinhança trata de 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 que é, uma especificação matemática, da relação entre as demais variáveis e a variável que desejamos maximizar ou minimizar.

Função objetivo é o objeto de nossa otimização. Pode ser um problema de otimização, um conjunto de teste para identificar os indivíduos mais aptos, ou mesmo uma "caixa preta" onde sabemos apenas o formato das entradas e a função nos retorna um valor que queremos otimizar. A grande vantagem dos algoritmos genéticos está no fato de não precisarmos saber como funciona esta função objetivo, apenas tê-la disponível para ser aplicada aos indivíduos e comparar os resultados.

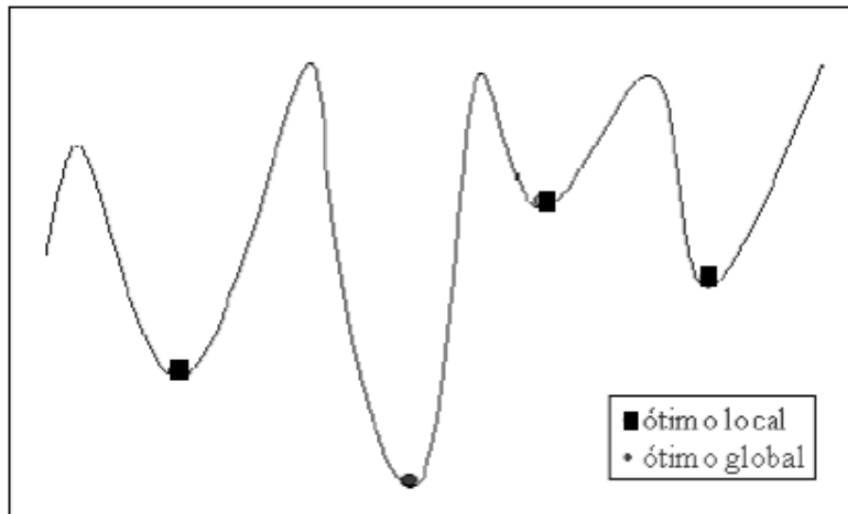
De acordo com a figura 1 pode-se observar a relação entre ótimo local e ótimo global em conjunto de soluções de um problema típico de minimização, os quadrados mostram soluções quase ótimas chamadas de ótimo local e o círculo mostra a melhor solução encontrada dentro do conjunto de soluções, no caso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

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

Conforme cita (RAUPP, 2003), o problema de otimização combinatória pode ser denominado como a ação de maximizar ou minimizar uma função objetiva de diversas variáveis sujeita a um conjunto de restrições, dentro de um contexto.

De acordo com (RAO, 1984) problemas do tipo POC tratam do estudo matemático para encontrar agrupamentos, arranjos ou a seleção ótima de objetos discretos, logo, não permitindo,

**Figura 1 – Representação de um problema de minimização com ótimos locais**



**Fonte: (TIMÓTEO, 2005)**

a utilização de métodos clássicos de otimização contínua para sua resolução.

Em seu trabalho (GOLBARG; LUNA, 2000) afirma que a ocorrência de problemas de otimização combinatória podem 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, cujo universo de dados é grande e 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 que compõem uma gama de soluções para problemas de otimização combinatória.

## 2.2 Heurística

O termo heurística é derivado do grego *heuriskein*, o que significa descobrir ou achar. De acordo com (TIMÓTEO, 2005) o significado da palavra em pesquisa operacional vai um pouco além da raiz etimológica. 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 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.

Ressalta-se que dentre as heurísticas, as chamadas 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 mais promissoras no espaço de soluções, além disto, as meta-heurísticas possuem grande abrangência, podendo ser aplicada à maioria dos problemas de otimização combinatória (NASCIMENTO; SILVA; ALVARENGA, 2005).

As meta-heurísticas surgiram como uma alternativa para amenizar a dificuldade que os métodos heurísticos tem de escapar dos chamados ótimos locais (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*).

### 2.2.1 Busca Tabu

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

Segundo (SUBRAMANIAN et al., 2006) BT 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  $S_0$ , o algoritmo explora cada iteração de um subconjunto  $V$  da vizinhança  $N(S)$  da solução corrente  $S$ . O membro  $S'$  de  $V$  com melhor valor nessa região segundo a função  $f(.)$  torna-



se a nova solução corrente mesmo que  $S'$  seja pior que  $S$  isto é, que  $f(S') > f(S)$  para um problema de minimização(SOUZA, 2000). A figura 20 que se encontra no apêndice representa o pseudocódigo do algoritmo da Busca Tabu.

Segundo (ARMENTANO; BRANCHINI, 2013) o algoritmo tem um intensivo uso de memória o que é uma característica essencial deste método. Para os autores o uso da memória pode ajudar a intensificar a busca em regiões com grande chances de se encontrar o resultado, ou até mesmo, diversificar a busca através de regiões inexploradas.

Ainda de acordo com (ARMENTANO; BRANCHINI, 2013) devemos adotar alguns procedimentos para que o processo de busca tenha um melhor resultado: listas tabu dinâmicas, passagens por regiões planas, intensificação, diversificação, *path relinking*. Listas tabu dinâmicas tem como objetivo evitar que o algoritmo entre em processo de ciclo. Passagens por regiões planas pode 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 são técnicas utilizadas 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 trata da intensificação de incorporar atributos de soluções de boa qualidade (chamadas de soluções elite), em seguida explora caminhos que contenham uma ou mais soluções de elite.

Ainda segundo (ARMENTANO; BRANCHINI, 2013) uma característica importante do método é que a solução final tem pouca ou nenhuma dependência da escolha feita para a solução inicial, isso graças aos mecanismos implementados pelo método, que fogem de ótimos locais.

### 2.2.2 Recozimento Simulado

Recozimento Simulado é a técnica de busca local probabilística, proposta originalmente por (KIRKPATRICK; JR.; VECCHI, 1983), que se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos.

Isto é, conforme (NORONHA, 2003) em analogia a física da matéria: levando um cristal a sua temperatura de fusão, as moléculas estão 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 possível.

Segundo (SOUZA; MARTINS; ARAÚJO, 2002) o processo se inicia com um membro qualquer do espaço de soluções, normalmente gerado aleatoriamente, e seleciona um de seus vizinhos randomicamente. Se este vizinho for melhor que o original ele é aceito e substitui a solução corrente. Se ele for pior por uma quantidade  $\Delta E$ , ele é aceito com uma probabilidade  $e^{-\Delta E/T}$ , onde T decresce gradualmente conforme o progresso 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 tomada como uma boa aproximação para a solução ótima. Originalmente, *Simulated Annealing* foi derivado de simulações em termodinâmica e por esta razão o parâmetro T é referenciado como temperatura e a maneira pela qual ela é reduzida é chamada de processo de resfriamento.

A figura 19 que se encontra no apêndice representa o pseudocódigo do algoritmo *Simulated Annealing*.

Conforme (AARTS; KORST, 1988) a analogia com a otimização (combinatória ou não) é bastante direta. Os estados da matéria são as soluções realizáveis, a quantidade objetiva substitui a energia, os estados metaestáveis da matéria sendo ótimos locais e a estrutura cristalina corresponde ao ótimo global.

### 2.2.3 Algoritmos genéticos

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

Para entender melhor (MITCHELL, 1998) descreve os principais termos biológicos necessários para o funcionamento dos algoritmos genéticos. Gene se trata de uma característica particular de um cromossomo. Um cromossomo é composto por um ou mais genes, pode se dizer também que é uma sequência de genes que será caracterizada como a solução do problema. *Fitness* significa a aptidão do indivíduo em um determinado ambiente. Indivíduo é a

combinação do cromossomo mais o *fitness* calculado através da função objetiva. 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, para que aconteça a evolução. Primeiramente é inicializada uma população, logo após iram acontecer os processos de seleção, reprodução também conhecida como *crossover* e mutação, os mesmos ocorreram a cada geração 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 um procedimento de busca no espaço de soluções potenciais para o problema e, como enfatiza (MICHALEWICZ; SCHOENAUER, 1996), esta busca requer um equilíbrio entre dois objetivos aparentemente conflitantes: a procura das melhores soluções na região que se apresenta promissora ou fase de intensificação e a procura de outra região ou exploração do espaço de busca, também conhecida como diversificação.

### 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, (LACERDA; CARVALHO, 1999) cita em seu trabalho que 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 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 é um método tradicional, para cada indivíduo é atribuído um espaço na roleta sendo o tamanho proporcional ao valor da aptidão do indivíduo. Esta roleta gira  $N$  vezes onde  $N$  é o número do tamanho da população, selecionando assim os pais para próxima geração. Seleção por torneio são selecionados  $X$  indivíduos da população onde  $X$  é um valor aleatório anterior e escolhidos os dois que contêm o maior valor de aptidão no caso os pais, segundo o autor este método 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 inversão, mutação e *crossover*, são responsáveis em realizar transformações nos indivíduos da população mas cada um possui suas diferentes funções dentro do algoritmo.

Inversão é um operador que modifica a genética de um gene ele é fundamental para garantir a biodiversidade da população, ainda segundo (GÓES, 2005) o operador mutação desenvolve o mesmo papel que o operador inversão, o mesmo cita que vários autores consideram inversão e mutação como o mesmo operador genético e também afirmam que são os operadores mais importantes e optam por trabalhar somente com estes operadores. Estes operadores são fundamentais para o desenvolvimento de algoritmos genéticos por evitarem a convergência prematura da solução, ou seja, quando uma população se estabiliza com uma adaptação pouco adequada, 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 onde é selecionado um ponto de corte produzindo duas cabeças e duas caldas 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).

No artigo apresentado pelos autores (LACERDA; CARVALHO, 1999) o elitismo é descrito como um operador genético que foi proposto por (JONG, 1975) em seu trabalho, que é dos pioneiros sobre algoritmos genéticos. Uma vez que os melhores indivíduos, de acordo com a função objetiva, podem ser perdidos entre uma geração e outra devido ao corte do *crossover* e a execução da mutação, torna-se interessante transferir o melhor indivíduo para a próxima

**Figura 3 – Ponto de corte *crossover***

$pai_1$	(0010101011 100000111111)
$pai_2$	(0011111010 010010101100)
$filho_1$	(0010101011 010010101100)
$filho_2$	(0011111010 100000111111)

**Fonte:** (LACERDA; CARVALHO, 1999)

geração, o nome dado para esta estratégia se chama elitismo uma técnica muito utilizada ao se desenvolver AGs. Os autores apresentam gráficos que mostram o desempenho da utilização do operador, segundo os mesmos 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.

### 2.3 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 acordo 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 a 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 resolvem este problema de forma manual, o que torna o processo árduo e demorado, podendo levar vários dias para ser concluído.

## 2.4 Trabalhos Relacionados

Foram encontrados vários trabalhos relacionados ao tema de resolução de problemas de otimização combinatória através de meta-heurísticas, os trabalhos relacionados escolhidos contem formas diferentes de resolução do mesmo problema o 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, o mesmo também afirma que o algoritmo apresentou-se robusto, uma vez que não houve grande variação na solução final, portando o mesmo sempre gerava soluções satisfatórias.

A forma abordada 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 trata da substituição do processo manual uma vez que se trata de varias instâncias.

A pesar da resolução do mesmo problema PAS através de diferentes técnicas meta-heurísticas podemos notar que todos trabalhos os algoritmos apresentam uma solução satisfatória segundo os autores.





todo início de semestre na instituição de ensino escolhida.

PQ do algoritmo genetico segundo renand dupas a escolha de algoritimos geneticos se dá através de uma comparação com outras técnicas e através disso de uma evidenciação das vantagens de sua utilização , logo o algoritmo genetico foi escolhido para este trabalho.

População inicial = –UMA NOVA ABORDAGEM PARA AUMENTAR A DIVERSIDADE.pdf

modelo matematico —APLICAÇÃO DE MODELO MATEMÁTICO, ABORDAGEM HEURÍSTICA E MÉTODO MISTO NA OTIMIZAÇÃO DA PROGRAMAÇÃO DE HORÁRIO DOS PROFESSORESTURMAS.pdf

restrições do problema e modelo matematico <http://www.dcc.ufla.br/infocomp/artigos/v4.3/art08.pdf>

Foi realizada uma análise dos requisistos atravez de entrevista com o stakeholder, após as entrevistas a modelagem de dados foi realizada de acordo com a demanda do projeto, para o desenvolvimento destes diagramas foi utilizadas a UML (Universal Modeling Language). Os seguintes diagramas foram desenhados, diagrama de caso de uso, diagrama de classe e diagrama de entidade relacionamento. Foram escolhidos os seguintes diagramas para que o sistema tenha uma documentação minima tendo em vista que o foco do trabalho é a resolução do problema de timetable através da utilização do algoritmo genetico.

Por se tratar de um sistema complexo, antes de iniciar a implementação do sistema fez-se necessário a sua modelagem. Segundo Elmasri Navathe (2005) as metodologias de modelagem de dados de objetos como UML (Universal Modeling Language – Linguagem de Modelagem Universal) estão se tornando cada vez mais populares no projeto e engenharia de software. Essas metodologias vão além do projeto de um banco de dados, especificando o projeto detalhado dos módulos de software e suas interações, utilizando vários tipos de diagramas.

### 3.1 Ferramentas Utilizadas

Este trabalho conta com a utilização de tecnologias proprias para o desenvolvimento de sistemas web, foram utilizadas as seguintes ferramentas: Para SGBD foi o utilizado PostgreSQL; No back-end foi utilizado Java e o *framework* Play!; No front-end as tecnologias utilizadas foram HTML, CSS, JavaScript e *framework* AngularJS e a IDE utilizada Eclipse.

### 3.1.1 Sistema de Gerenciamento de Banco de Dados

O SGBD escolhido foi o PostgreSQL pelo fato de ser uma ferramenta open-source e que trabalha perfeitamente com o framework escolhido Play!, uma vez que utilizado em projetos anteriores não foram apresentados conflitos entre o framework e o SGBD. A seguir pode ser notar que é uma ferramenta robusta e que tem visão no mercado internacional.

O PostgreSQL é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura que comprovadamente ganhou forte reputação de confiabilidade, integridade de dados e conformidade a padrões. Roda em todos os grandes sistemas operacionais. É totalmente compatível com ACID, tem suporte completo a chaves estrangeiras, junções (JOINS), visões, gatilhos e procedimentos armazenados (em múltiplas linguagens). Inclui a maior parte dos tipos de dados do ISO SQL:1999, incluindo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP. Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos. Possui interfaces nativas de programação para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros, e uma excepcional documentação.(POSTGRESQL, 2013)

Para a resolução do problema de timetable foi escolhido o algoritmo genético, a escolha do algoritmo foi devida a grande utilização do mesmo para resolução de problemas do tipo NP-difícil que foram encontrados na literatura.

### 3.1.2 Ferramentas Back-end

Foi escolhida uma linguagem de programação Java por ser orientada a objeto. Também foi escolhido o *Play! framework*, para que o desenvolvimento aconteça de forma mais rápida, fácil e eficiente.

Java foi criada pela Sun Microsystems para desenvolver inovações tecnológicas em 1992, time liderado por James Gosling. O Java utiliza do conceito de máquina virtual, onde existe, entre o sistema operacional e a aplicação, uma camada extra responsável por traduzir mas não apenas isso - o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional, onde ela está rodando no momento. Sua aplicação roda sem nenhum envolvimento com o sistema operacional, sempre conversando apenas com a JVM - Java Virtual Machine

(CAELUM, 2013).

Em 2009 a Oracle comprou a Sun, fortalecendo a marca. A Oracle sempre foi, junto com a IBM, uma das empresas que mais investiram e fizeram negócios através do uso da plataforma Java. Em 2011 surge a versão Java 7 com algumas pequenas mudanças na linguagem (CAELUM, 2013).

The Play! É um moderno framework MVC de alta produtividade, que utiliza Java e Scala para o desenvolvimento web, open-source , utiliza templates, hibernate e JUnit em sua arquitetura. Existe duas versões do framework Play! 1 e Play2! este trabalho utiliza a versão 1 do framework(PLAY!, 2013).

### 3.1.3 Ferramentas Front-end

As ferramentas de Front-end descritas abaixo, foram escolhidas devida a grande utilização na web grande parte dos sites contem HTML, CSS ou JavaScript em algum trexo de seu código, foi escolhido tambem o framework AngularJS para que o desenvolvimento ocorra de maneira agil e mais rapida.

HTML que é defindo por (*HyperText Markup Language*) ou linguagem de marcação, é uma linguagem que é utilizada no desenvolvimento de paginas web (W3C, 2013 a).

Cascading Style Sheets (CSS) é uma tecnologia utilizada para adicionar estilos como cores, fontes, espaçamentos em documentos escritos em uma linguagem de marcação como exemplo o HTML (W3C, 2013 b).

JavaScript é uma linguagem de script utilizada no desenvolvimento de paginas na web, atualmente é a principal linguagem para programação client-side em navegadores web. Todas as paginas de HTML modernas estão usando JavaScript para adicionar funcionalidades e para se comunicar com os webServers(W3SCHOOLS, 2013).

Angularjs é um *framework JavaScript* construido e mantido pelo grupo de engenheiros do Google, ele usa o HTML como uma *template engine*, tudo isso no intuito de fornecer uma solução completa para o cliente-side de sua aplicação. Além disso tem total compatibilidade com as bibliotecas javascript mais utilizadas, como jQuery. É um novo conceito para desenvolvimento de web apps client-site.(MENDES, 2013)

### 3.1.4 IDE

O Eclipse é uma IDE (*integrated development environment*). Diferente de uma RAD(*Rapid Application Development*), onde o objetivo é desenvolver o mais rápido possível através do arrastar-e-soltar do mouse, onde montanhas de código são gerados em background, uma IDE te auxilia no desenvolvimento, evitando se intrometer e fazer muita mágica (CAELUM, 2013).

O Eclipse é a IDE líder de mercado. Formada por um consórcio liderado pela IBM, possui seu código livre. A última versão é a 4.3, mas com qualquer versão posterior a do 3.1 você terá suporte ao Java 5, 6 e 7 (CAELUM, 2013).

Esta IDE foi escolhida devido ao grande reconhecimento mundial, por sua eficiência ao se trabalhar com a linguagem de programação Java, por ser open-source e pela existência de várias ferramentas criadas pela comunidade, para o auxílio no desenvolvimento de softwares.

## 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 definida como NP-Difícil, logo a resolução manual de problemas desse tipo, se torna inviável em um espaço de tempo curto, para isto são utilizadas meta-heurísticas que são métodos de aproximação, que ajudam a encontrar uma solução em um conjunto de soluções viáveis. Este trabalho utiliza informações fornecidas pela FAFICH UFMG, nestas informações estão contidas salas com suas propriedades, e disciplinas enviadas pelos colegiados com suas restrições de horários e informações básicas de cada uma. A resolução do problema PAS consiste então na alocação de todas as disciplinas enviadas pelos colegiados 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

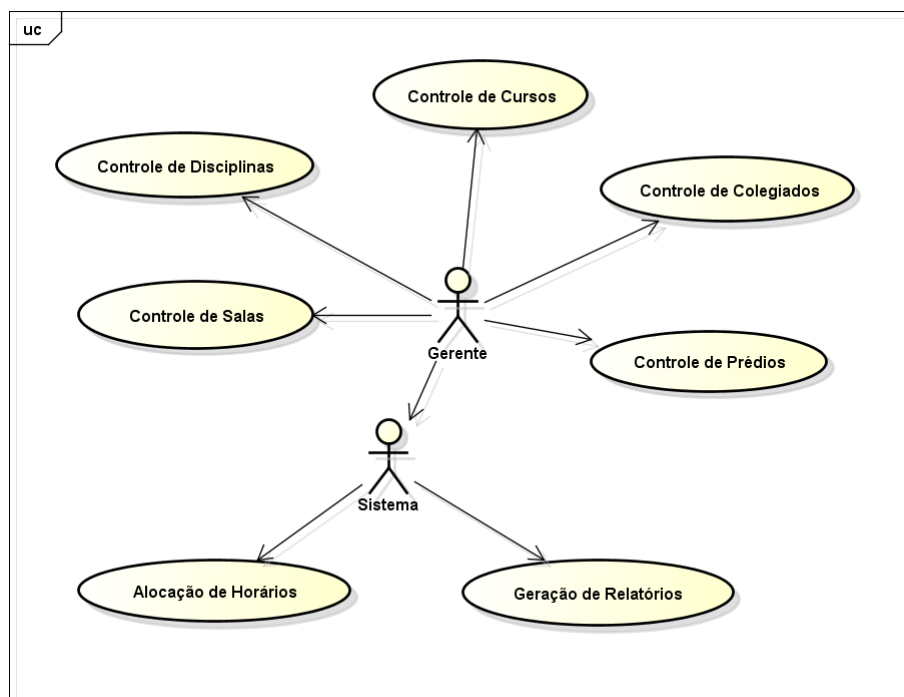
#### 4.1.1 Diagramas de caso de uso

A Figura XX descreve todas as funcionalidades que o sistema possui, essas funcionalidades foram divididas em 2 atores "Gerente" e "Sistema" cada um 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.

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.

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

**Figura 6 – Diagrama de Caso de Uso**



**Fonte: Desenvolvido pelo autor**

algoritmo genético para encontrar a melhor solução do problema e a funcionalidade geração de relatórios mostra para o "Gerente" o melhor resultado de alocação encontrado pelo algoritmo.

#### 4.1.2 Diagrama de Entidade Relacionamento

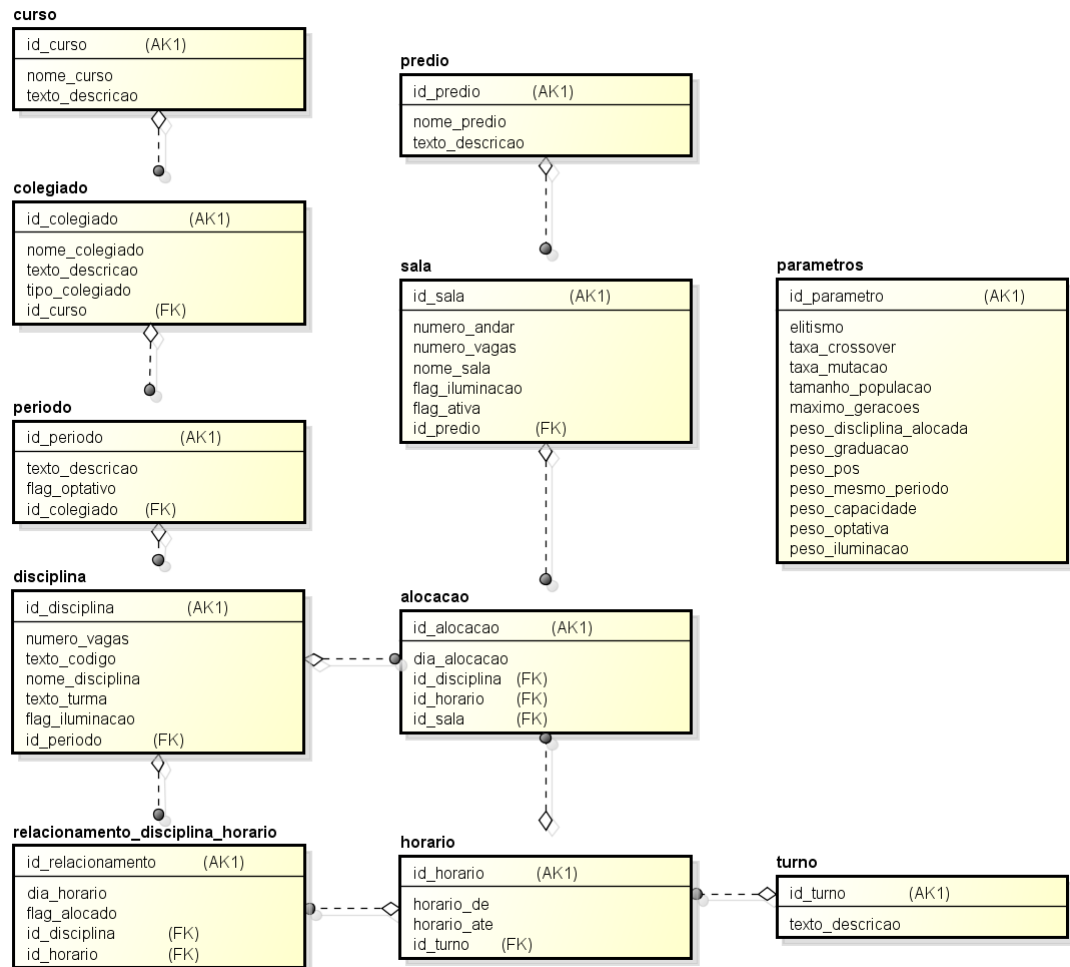
Na Figura XX mostra o relacionamento das tabelas no sistema através do diagrama de entidade relacionamento.

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 como nulo o que significa que em um específico horário e em uma determinada sala não existe

**Figura 7 – Diagrama de Entidade Relacionamento**



Fonte: Desenvolvido pelo autor

disciplina alocada.

A tabela parâmetro é utilizada apenas para guardar os últimos parâmetros utilizados na execução do algoritmo genético.

#### 4.1.3 Diagramas de classe

Este diagrama é das classes do algoritmo genético, as classes do sistema em geral não serão abordados. Quando terminar o código revisar e explicar.





### 4.2.1 Indivíduo

O fitness do indivíduo é calculado através da função objetivo que será abordada em outro tópico, brevemente explicando se trata de um valor que vai de 0 a 100 onde 100 é a pontuação máxima do indivíduo está nota é alcançada quando todos os requisitos de alocação desejados foram atendidos.

O termo gene representado pela figura XX possui uma combinação de quatro variáveis sala, dia da semana, horário e o relacionamento de obrigatoriedade "disciplina horário". As variáveis sala, horário e dia da semana são fixas, 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 específica 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: Desenvolvido pelo autor**

Um 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 indivíduo contém todas as variáveis relacionamentos de obrigatoriedade nulas, 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, esta alocação é medida pela sua pontuação de fitness, podendo ser ou não o resultado do problema. Os valores utilizados na representação do cromossomo na figura XX 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.

O termo indivíduo é então composto pelo cromossomo juntamente com a pontuação adquirida após a execução do método de cálculo de fitness, a figura XX demonstra a representação de um indivíduo.

**Figura 10 – Representação Cromossomo**

Cromossomo							
1	null	null	3	4	null	...	7

Fonte: Desenvolvido pelo autor

**Figura 11 – Representação Indivíduo**

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

Fonte: Desenvolvido pelo autor

#### 4.2.2 População

Uma população, quando relacionada aos termos genéticos se trata de um conjunto de indivíduos, também representa uma interação do algoritmo genético, ou seja uma geração. 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 atinga o tamanho da população que deve ser igual ao parâmetro tamanho da população.

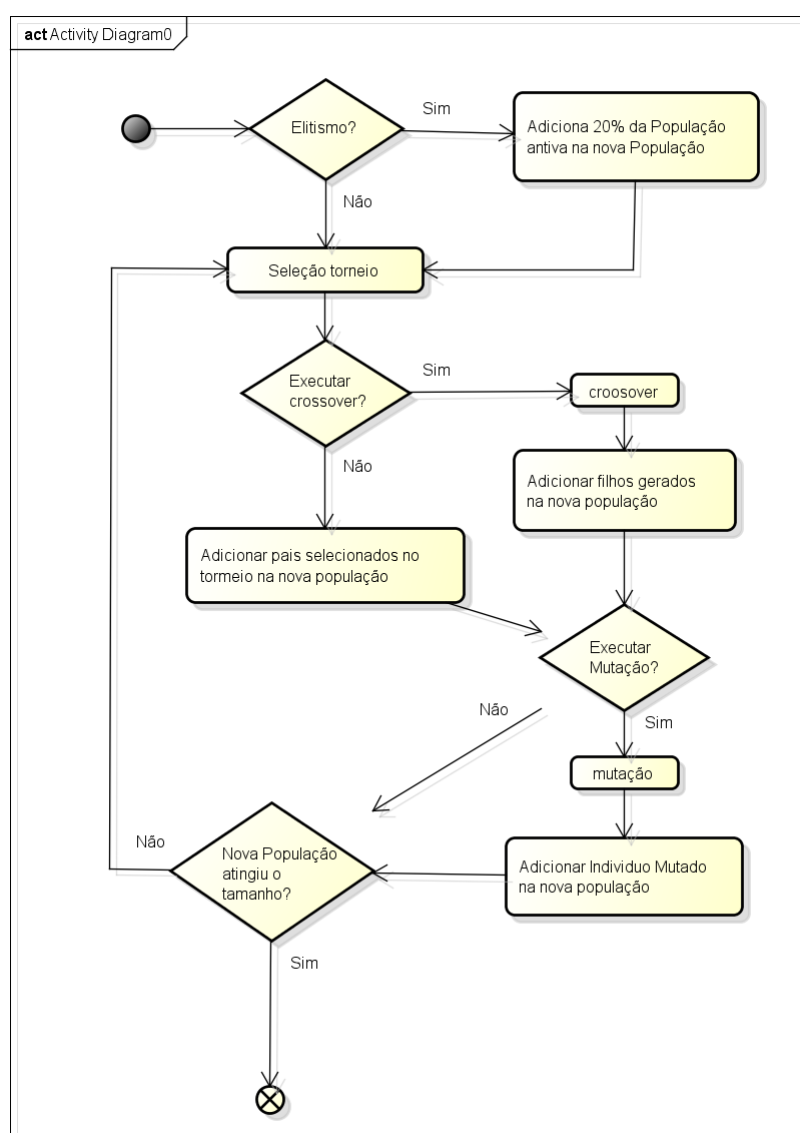
Para cada geração, é criada uma nova população a partir da população criada na geração anterior, se o operador genético elitismo estiver com o valor verdadeiro, iniciamos esta nova população com 20% dos melhores indivíduos da população anterior, os melhores indivíduos de uma população são indicados pelas maiores pontuações 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 randomicos 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 denominado 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 de uma combinação, logo sem seguida serão adicionados na nova população, em caso de falso os pais serão os indivíduos adicionados na nova população.

Para se utilizar a operação de mutação novamente é utilizado outro valor randominco se a condição for verdadeira um indivíduo da população anterior é selecionado e o mesmo sofrerá a mutação genética, o individuo 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 é descrito na figura XX.

**Figura 12 – Fluxo Nova População**



Fonte: Desenvolvido pelo autor

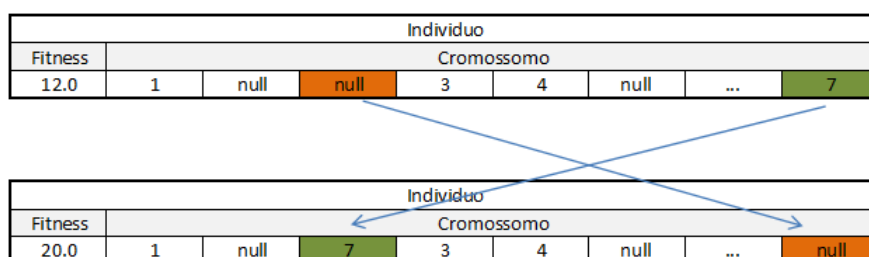
### 4.2.3 Operadores genéticos

Um dos operadores genéticos utilizados neste trabalho conforme citado anteriormente é o elitismo, este operador genético com valor verdadeiro garante a seleção 20% dos melhores indivíduos para a próxima geração.

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

Primeiramente são escolhidos de forma randômica dois genes do cromossomo, apos a escolha dos genes a serem trocados, é feita a troca dos genes de posição, no caso apenas a relação de obrigatoriedade é trocada, permanecendo as outras propriedades, então retornado um novo indivíduo que tem a composição genética alterada. Apos 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.

**Figura 13 – Representação Mutação**



Fonte: Desenvolvido 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 os dois que contem a maior pontuação de *fitness* são selecionados e enviados para o *crossover*.

Para o método de *crossover* inicialmente é escolhido um ponto de corte para se realizar o cruzamento entres os pais (pai e mãe). A figura XX apresenta os indivíduos antes do cruzamento e o indivíduo pai marcado com o ponto de corte.

Após a escolha o filho 1 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 nulos quando criados, quando ocorre a troca genética estamos

**Figura 14 – Indivíduos antes do crossover**

Cromossomo Pai									
1	null	3	null	4	null	2	7	8	6

Cromossomo Mãe									
4	null	1	8	7	null	6	4	3	null

Cromossomo Filho									
null	null	null	null	null	null	null	null	null	null

Fonte: Desenvolvido pelo autor

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 no ponto de corte são removidos do indivíduo mãe. A marca no cromossomo filho representa o ponto de corte. Conforme apresentado na figura XX.

**Figura 15 – Inserção do material genético do pai**

Cromossomo Pai									
1	null	3	null	4	null	2	7	8	6

Cromossomo Mãe						
4	8	7	null	6	4	null

Cromossomo Filho									
1	null	3	null	null	null	null	null	null	null

Fonte: Desenvolvido pelo autor

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

**Figura 16 – Inserção do material genético da mãe**

Cromossomo Pai									
1	null	3	null	4	null	2	7	8	6

Cromossomo Mãe						
4	8	7	null	6	4	null

Cromossomo Filho									
1	null	3	4	8	7	null	6	4	null

Fonte: Desenvolvido pelo autor

Este mesmo processo é realizado para a criação do filho 2 porem, ao criar este novo filho, devemos trocar as operações realizadas entre o pai e a mãe invertendo os mesmo de lugar no fluxo descrito.

Após a realização do *crossover* os filhos recebem uma nova nota de fitness que conforme dito anteriormente pode ser maior ou menor do que a anterior.

#### 4.2.4 Definição da função objetivo

O calculo da função objetivo é realizado através da regra 3 para calcular cada restrição. Para o calculo da primeira restrição chamada de fitness01 que se trata da restrição de obrigatoriedade entre as disciplinas e horários, é realizado um somatório de todos relacionamentos que estão de acordo com o gente em que estão alocados em é aplicada a regra de 3 em cima do número de registros contidos na tabela de relacionamento entre as disciplinas e horários contDisciplinaHorario.

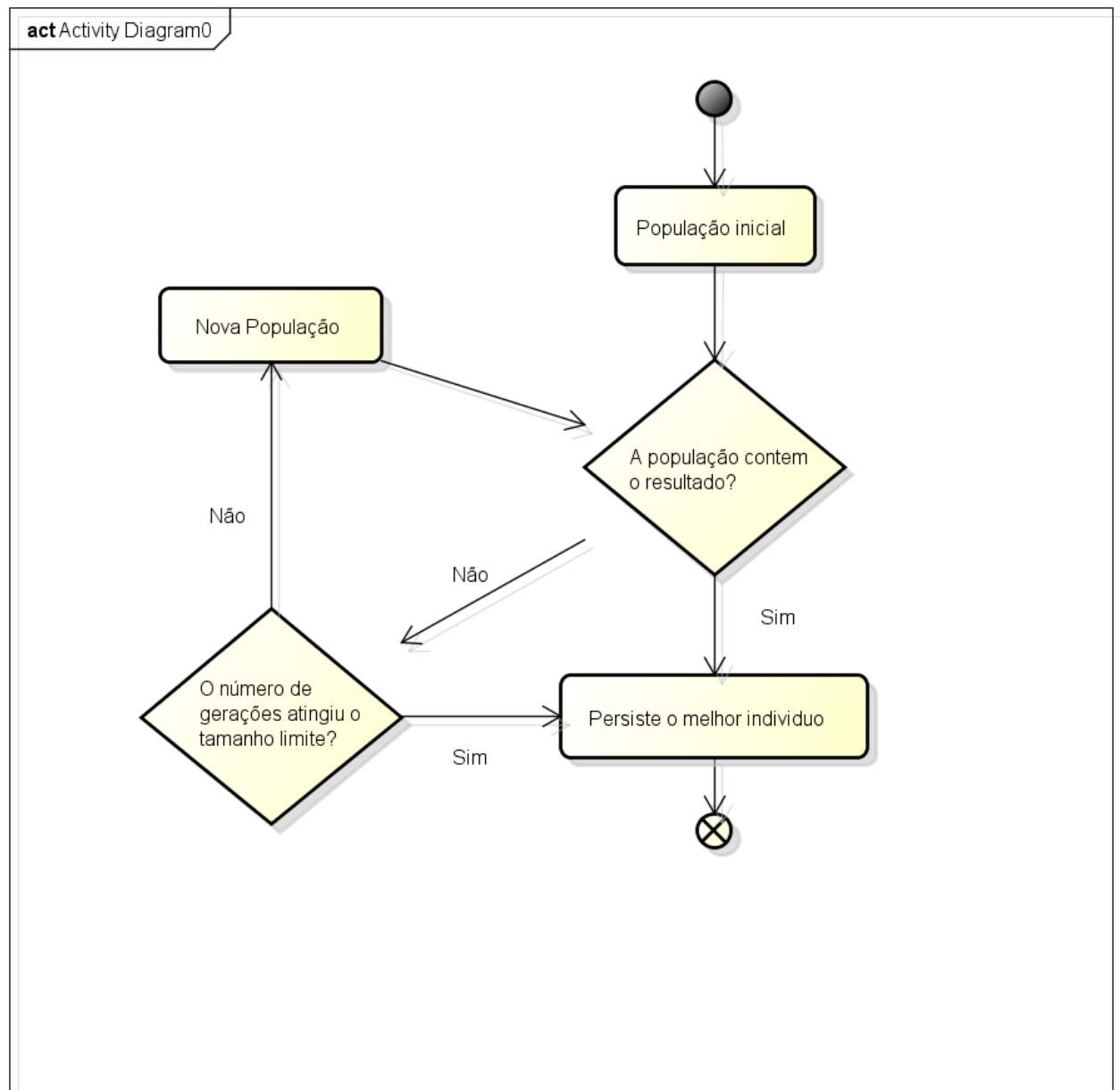
$$fitness01 = \sum_{i=1}^n *100/contDisciplinaHorario$$

O mesmo calculo é realizado para as outras restrições disciplinas que os horários estão na mesma sala que é chamado de fitness02, e capacidade da sala com a capacidade necessária pela disciplina fitness03.

O fitness final é calculado pela formula (fitness01+fitness02+fitness03)/3 este calculo tem como valor máximo 100 e minimo 0.

#### 4.2.5 Fluxo do algoritmo

O fluxo do algoritmo conforme a imagem XX é iniciado pela criação da população inicial, para cada interação do algoritmo é verificado se a população contem 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 de acordo coma figura XX

**Figura 17 – Fluxo do algoritmo**

Fonte: Desenvolvido pelo autor

## 5 RESULTADOS OBTIDOS

Após o sistema implementado foram realizadas uma bateria de testes para que os valores parâmetros iniciais fossem definidos, os dados utilizados tem em sua composição o total de 10 salas e um curso com o total de 46 disciplinas e duas respectivas obrigаторiedades, serão utilizados como parâmetros iniciais os dados representados na Figura 18, no caso uma população com 100 indivíduos, 60% de crossover 20% de mutação, e elitismos estes dados foram tirados como base na conclusão do trabalho de (CISCON, 2006). Os testes realizados tiveram uma máquina com a seguinte configuração, Intel(R) Core(TM) i5 3.40GHz com 16Gb de memória RAM sob o sistema operacional Windows 7 64bit.

**Figura 18 – Parâmetros utilizados para o teste inicial**

Parâmetros Algoritmo

Crossover: 0.6

Mutação: 0.3

Tamanho População: 140

Número de Gerações: 500

Elitismo: ☒ Sim ☐ Não

**Fonte: Desenvolvido pelo autor**

Gráfico com todos os parâmetros

Gráfico sem crossover

Gráfico sem mutação

Gráfico sem elitismo

Tempo de cada geração

Depois do sistema implementado

Entrada processamento e saída

ajuste do algoritmo de alocação

falar do algoritmo que não evolui sem o crossover so com mutação.



## **6 CONSIDERAÇÕES FINAIS**

\*Relembra ro problema e comprar como o resultadoo bjetido.

Discussão dos resultados obtidos na pesquisa, onde se verificam as observações pessoais do autor. Poderá também apresentar sugestões de novas linhas de estudo. A conclusão deve estar de acordo com os objetivos do trabalho. A conclusão não deve apresentar citações ou interpretações de outros autores.

### **6.1 Trabalhos futuros**

Criar um DW para geração dos relatorios de acordo com a dimensão escolhida.

Utilização de outros algoritimos para a resolução do problema ex. algoritimos evolutivos formiga entre outros.

Pegar o feed back do usuario para melhoria na interface, e do algoritimo.

## 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. *See also*: <http://www.literature.org/authors/darwin-charles/the-origin-of-species>, 1968.
- 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 flow problems. 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 simulated 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. *Monografia de Graduação*, 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.
- MENDES, W. *AngularJS um framework para facilitar sua vida*. 2013. Disponível em: <<http://www.slideshare.net/WilsonMendes/angularjs-um-framework-para-facilitar-sua-vida>>. Acesso em: 29 set. 2013.
- 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. *Projeto Orientado—Universidade Federal do Rio Grande do Norte. Rio Grande do Norte*. Disponível em: <http://www.sbc.org.br/reic/edicoes/2001e1/cientificos/UmaAbordagemSobreEstrategiasMetaheurísticas.pdf>, 2003.
- OLIVEIRA, A. C. de. Uso do algoritmo genético e recozimento simulado para o problema de alocação de salas. *Monografia, 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, 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>>.

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, 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.; ARAÚJO, 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.

TIMÓTEO, 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.

[titletoc,toc,page]appendix

**Figura 19 – Representação do pseudocódigo do algoritmo da Busca Tabu**

```

procedimento  $BT(f(\cdot), N(\cdot), A(\cdot), |V|, f_{min}, |T|, BTmax, s)$ 
1   $s^* \leftarrow s;$            {Melhor solução obtida até então}
2   $Iter \leftarrow 0;$        {Contador do número de iterações}
3   $MelhorIter \leftarrow 0;$  {Iteração mais recente que forneceu  $s^*$ }
4   $T \leftarrow \emptyset;$    {Lista Tabu}
5  Inicialize a função de aspiração  $A$ ;
6  enquanto  $(f(s) > f_{min} \text{ e } Iter - MelhorIter < BTmax)$  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     $T \leftarrow T - \{\text{movimento mais antigo}\} + \{\text{movimento que gerou } s'\};$ 
10   Atualize a função de aspiração  $A$ ;
11    $s \leftarrow s';$ 
12   se  $(f(s) < f(s^*))$  então
13      $s^* \leftarrow s;$ 
14      $MelhorIter \leftarrow Iter;$ 
15   fim-se;
16 fim-enquanto;
17  $s \leftarrow s^*;$ 
18 Retorne  $s$ ;
fim  $BT$ ;

```

Fonte: (SOUZA, 2000)

**Figura 20 – Representação do pseudocódigo do algoritmo *Simulated Annealing***

```

procedimento  $SA(f(\cdot), N(\cdot), \alpha, SAmaz, T_0, s)$ 
1   $s^* \leftarrow s;$            {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  $(IterT < SAmaz)$  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  $(f(s') < f(s^*))$  então  $s^* \leftarrow s';$ 
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   $s \leftarrow s^*;$ 
22 Retorne  $s$ ;
fim  $SA$ ;

```

Fonte: (SOUZA; MARTINS; ARAÚJO, 2002)