



FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Jéssica Marrero Troncoso

**Algoritmo Genético aplicado a Geração Automática de Grade
de Horários**

Americana, SP

2016



FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Jéssica Marrero Troncoso

**Algoritmo Genético aplicado a Geração Automática de Grade
de Horários**

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de sistemas sob a orientação do Prof. Me. Kleber de Oliveira Andrade

Área de concentração: Inteligência Artificial

Americana, SP.

2016

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

T767a TRONCOSO, Jéssica Marrero
Algoritmo genético aplicado a geração automática de grade de horários. / Jéssica Marrero Troncoso. – Americana: 2016.
46f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.

Orientador: Prof. Ms. Kléber de Oliveira Andrade

1. Inteligência artificial IA. ANDRADE, Kléber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 007.52


Jessica Marrero Troncoso

ALGORITMO GENÉTICO APLICADO A GERAÇÃO AUTOMÁTICA DE GRADE DE HORÁRIOS


Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Inteligência Artificial.

Americana, 08 de dezembro de 2016.


Banca Examinadora:



Kléber de Oliveira Andrade (Presidente)
Mestre
Fatec Americana



Alberto Martins Júnior (Membro)
Mestre
Fatec Americana



Paula da Fonte Sanches (Membro)
Mestre
Fatec Americana

AGRADECIMENTOS

A minha família por todo apoio e ajuda durante meus estudos.

Ao meu orientador professor Me. Kleber de Oliveira Andrade, por toda paciência, ensinamentos, dedicação e contribuições.

A meu namorado Ricardo pela paciência, ajuda e contribuições para a realização deste trabalho.

A todos os amigos e companheiros que direta ou indiretamente me ajudaram. Em especial: Diana Bernardo, Luiz Carlos Junior, Jacomo Giovanetti, Natália Rocha e Viviane Maringolo.

RESUMO

Este trabalho apresenta a solução de um problema muito comum nas instituições de ensino, o problema de distribuição de grade horária. Problemas de grade horária são complexos devido ao grande número de variáveis envolvidas, além de possuir diversas soluções possíveis. Para solucionar este problema foi utilizado o algoritmo genético. Algoritmo Genético é um ramo da inteligência artificial que possibilita uma busca heurística em um grande espaço de soluções possíveis. Ele se baseia na evolução natural, onde a cada geração de uma população, a mesma evolui e se adapta ao ambiente ao qual ela está inserida. Partindo desse princípio podemos utilizá-lo para resolver diversos problemas complexos. Ele possui uma estrutura genérica, onde se modela o cromossomo/indivíduo de acordo com o problema, avalia o mesmo, seleciona os mais aptos (de acordo com a avaliação), utiliza operadores de cruzamento aos indivíduos “pais” para que possam gerar “filhos” melhores e que terão mais chances de sobrevivência na população, realiza-se a mutação e o avalia novamente, repetindo o processo até que se atinja um critério de parada predefinido. Essa estrutura pode e deve ser modelada de acordo com o problema. Para a resolução do problema de grade horária proposto neste trabalho, o cromossomo foi estruturado de forma que já suprisse algumas restrições, fazendo com que fosse apenas necessário alocar o horário de aula na mesma. Utilizando uma avaliação onde caso o indivíduo que infringisse alguma restrição sofresse uma penalidade (valor do *fitness*), os indivíduos com um menor valor de *fitness* estão mais próximos de uma solução para o problema. Foi desenvolvido um software para realizar a distribuição utilizando o AG e para a codificação a linguagem Java e o banco de dados em MySQL. Foi possível observar na execução do algoritmo que a cada geração a população evoluía, e que o AG foi capaz de trazer uma boa solução para o problema.

Palavras Chave: algoritmo genético, software, *timetabling*

ABSTRACT

This work presents the solution of a problem very common in educational institutions, the problem of grid distribution. Time grid problems are complex due to the large number of variables involved, besides having several possible solutions. To solve this problem we used the genetic algorithm. Genetic Algorithm is a branch of artificial intelligence that enables a heuristic search in a large space of possible solutions. It is based on natural evolution, where each generation of a population, it evolves and adapts to the environment where it is inserted. This principle can be used to solve several complex problems. It has a generic structure, where its chromosome is molded then evaluated, selects the fit ones (according to the evaluation), uses crossover operators to the chromosome "parents" so that they can generate better "children" and they will have more chances of survival in the population. The mutation is performed and then the chromosome is evaluated again, repeating the process until a predetermined stopping criterion is reached. This structure can and should be modeled according to the problem. In order to solve the problem of the time grid proposed in this work, the chromosome was structured in such a way to satisfy some restrictions, being only necessary to allocate the class schedule in it. If any individual violated any restrictions he would suffer a penalty (fitness value increased), so individuals with a lower fitness value are closer to a solution to the problem. The software was developed to perform the time grid distribution applying GA, Java language for encoding and MySql for the database. It was possible to observe in the execution of the algorithm that each generation the population evolved, and that the GA was able to bring a good solution to the problem.

Keywords: *genetic algorithm, software, timetabling.*

SUMÁRIO

1. INTRODUÇÃO	1
2. OTIMIZAÇÃO DE GRADE HORÁRIA	2
2.1. <i>TIMETABLING</i>	2
2.2. O PROBLEMA DA GRADE DE HORÁRIOS	5
3. ALGORITMO GENÉTICO	7
3.1. FUNCIONAMENTO GENÉRICO DO ALGORITMO	9
3.1.1. REPRESENTAÇÃO GENÉRICA DO CROMOSSOMO	10
3.1.2. A POPULAÇÃO INICIAL.....	11
3.1.3. AVALIAÇÃO DE INDIVÍDUOS.....	11
3.1.4. ELITISMO	12
3.1.5. MÉTODOS DE SELEÇÃO	12
3.1.6. OPERADOR DE <i>CROSSOVER</i>	13
3.1.7. OPERADOR DE MUTAÇÃO.....	15
3.1.8. NOVA POPULAÇÃO	16
4. ALGORITMO GENÉTICO PROPOSTO.....	17
4.1. O CROMOSSOMO.....	17
4.2. POPULAÇÃO INICIAL.....	18
4.3. AVALIAÇÃO	18
4.4. ELITISMO.....	19
4.5. MÉTODO DE SELEÇÃO ROLETA.....	19
4.6. CRUZAMENTO (<i>CROSSOVER</i>).....	21
4.7. MUTAÇÃO.....	21
4.8. CRITÉRIO DE PARADA.....	22
5. IMPLEMENTAÇÃO DO SISTEMA.....	23

5.1.FERRAMENTAS UTILAZADAS	23
5.2.DIAGRAMA DE CLASSES.....	23
5.3.DIAGRAMA DE ENTIDADE RELACIONAMENTO.....	25
6. EXPERIMENTOS E RESULTADOS	31
7. CONSIDERAÇÕES FINAIS	33
REFERENCIAS BIBLIOGRAFICAS	34
APÊNDICE	36

LISTA DE ABREVIATURAS

AG - Algoritmo Genético

BD - Banco de Datos

IDE - *Integrated Development Environment*

SGBD - Sistema Gerenciador de Banco de Datos

LISTA DE TABELAS E QUADROS

Tabela 1 - Exemplo de dados para o método da roleta.....	20
Quadro 1 - Tabela BD - Atribuição	27
Quadro 2 - Tabela BD - Aula.....	27
Quadro 3 - Tabela BD - Curso.....	27
Quadro 4 - Tabela BD - CursoPeriodo	28
Quadro 5 - Tabela BD - CursoPeriodoHorario.....	28
Quadro 6 - Tabela BD - Disciplina.....	28
Quadro 7 - Tabela BD - Disponibilidade.....	29
Quadro 8 - Tabela BD - Horario	29
Quadro 9 - Tabela BD - Periodo.....	29
Quadro 10 - Tabela BD - Professor.....	29
Quadro 11 - Tabela BD - Semestre.....	30
Quadro 12 - Tabela BD - Turma.....	30

LISTA DE GRÁFICOS

Gráfico 1 - Roleta de Probabilidade de Seleção.....	20
Gráfico 2 - Simulação AG em 100 gerações	31

LISTA DE FIGURAS

Figura 1 - Exemplo N-Rainhas	8
Figura 2 - Representação do Algoritmo Genético	10
Figura 3 - <i>Crossover</i> um ponto de corte	14
Figura 4 - <i>Crossover</i> dois pontos de corte	14
Figura 5 - <i>Crossover</i> unilateral	15
Figura 6 - Estrutura do Cromossomo proposto	17
Figura 7 - <i>Crossover</i> por ordem exemplo	21
Figura 8 - Exemplo de Mutação	22
Figura 9 - Diagrama de Classes Algoritmo Genético	24
Figura 10 - Diagrama de Classes – Classes de Persistência.....	25
Figura 11 - Diagrama Entidade Relacionamento.....	26

1. INTRODUÇÃO

As instituições de ensino no início de todo ano letivo possuem a necessidade da elaboração de uma grade de horários na qual serão definidos os horários de aulas, tanto para alunos, quanto para docentes. Este problema de escalonamento de horários é um problema clássico que tem como objetivo alocar eventos (aulas, avaliações e afins) de acordo com os recursos oferecidos pela instituição.

A grade horária tem muitas variáveis o que a torna um problema complexo para ser resolvido manualmente. Geralmente este problema é resolvido com programação linear, mas também podem ser utilizados heurísticas. Uma das heurísticas comumente utilizada é o Algoritmo Genético (AG). O funcionamento deste algoritmo é baseado na evolução genética na qual apenas os melhores indivíduos de uma população se reproduzem para gerar filhos (nova população) cada vez mais adaptada.

Este trabalho tem como objetivo geral desenvolver e implementar um AGs para gerar automaticamente a distribuição da grade horária de uma instituição de ensino. Para isso, será necessário utilizado a linguagem Java e o banco de dados MySQL. Como estudo de caso, será utilizado a grade de horários de uma Instituição de Ensino Superior que tem 7 cursos, cada curso tem muitas disciplinas distribuídas em 6 semestres cada. Essas disciplinas são ministradas por diversos professores em 3 períodos diferentes (matutino, vespertino e noturno).

Esta monografia está dividida em 7 capítulos. O Capítulo 2 descreve o problema da grade horária conhecido como *timetabling*. No Capítulo 3 é descrito o funcionamento de um AG básico, explicando como cada passo funciona. O Capítulo 4 propõe um AG para a resolver o problema da grade de horários. O Capítulo 5 explica a implementação do sistema e as ferramentas utilizadas. O Capítulo 6 detalha os experimentos realizados e discute as soluções. Por último, o capítulo 7 é reservado para as considerações finais e trabalhos futuros.

2. OTIMIZAÇÃO DE GRADE HORÁRIA

O problema da grade de horários é muito conhecido nas instituições de ensino por conter um grande número de variáveis e restrições. Sendo assim este problema tem um espaço de busca geralmente grande, o que o torna complexo para ser resolvido de forma manual. Braz Júnior (2000) comenta que a grade horária é uma relação entre vários elementos, como tempo, professores, turmas e disciplinas. Estes elementos são chamados de recursos, suas características são especificadas pelo problema, sendo a relação entre eles o que determinará a solução para o problema.

Dentre diversos recursos que podem ser otimizados em uma instituição de ensino está a disponibilidade dos docentes e de salas e laboratórios de aulas. Otimizar estes recursos permite que os docentes sejam beneficiados com tempo para realizar outras atividades, tanto pessoais, quanto profissionais, desta forma até enriquecendo sua formação (ANDRADE, 2014).

Este tipo de problema pertence a um conjunto de problemas de otimização conhecidos como *timetabling*. Este tipo de problema consiste em otimizar os recursos distribuídos em uma determinada estrutura de horários fixos para realização de atividades, que não precisam ser apenas educacionais, podem ser por exemplo de escalonamento de horários de um hospital, distribuição de energia elétrica, planejamento de transporte público entre outros, porém o problema de *timetabling* é mais abordado para elaboração de grades horárias escolares, de faculdades e afins.

2.1. TIMETABLING

De acordo com Borges (2003) a essência do *timetabling* é a problematização de escalonamento, na qual o problema se sujeita a restrições e cada evento é considerado como instância, ou recurso do problema ou um conjunto de eventos que necessitam acontecer em um período finito de tempo. Sendo assim, um recurso não pode ser solicitado por mais de um evento ao

mesmo tempo e devem existir recursos suficientes para atender todos os eventos durante o processo de escalonamento.

Um *timetabling* pode ser genericamente representado por três conjuntos básicos, conforme Borges (2003 apud Ross (*et al.* 2000)):

- $A = \{a_1, a_2, \dots, a_n\}$, ou seja, um conjunto finito de atividades diversas como exames, seminários, projetos ou aulas;
- $T = \{t_1, t_2, \dots, t_n\}$, que é um conjunto finito de horários, para realização das atividades, e
- $R = \{r_1, r_2, \dots, r_n\}$, que implica em um conjunto finito de recursos (instrutores, monitores ou professores), que têm o papel de monitorar eventos particulares.

Então é válido representar um conjunto de elementos de *timetabling* como $\{a, t, r\}$ onde $a \in A$ (conjunto de atividades), $t \in T$ (conjunto de horários) e $r \in R$ (conjunto de recursos), podendo ser interpretado como: a atividade a se inicia em um tempo t e tem como recursos r . Portanto, *timetabling* é uma coleção de conjuntos, um para cada atividade, em que as escalas geradas não podem infringir um conjunto de restrições pré-definidos para o problema.

Uma distribuição genérica de *timetabling* está condicionado a diversas restrições, sendo duas delas as principais para que se obtenha um padrão de qualidade ou regras para a geração das tabelas de escalonamento, sendo elas:

- i. Nenhum recurso pode ser utilizado em mais de um local ao mesmo tempo
- ii. O recurso solicitado pelas atividades dentro do tempo determinado não pode ultrapassar os recursos disponíveis.

As restrições variam de acordo com cada problema e dependendo do objetivo, ao se violar uma restrição, as tabelas geradas não poderão ser aplicadas. Os principais tipos de restrições podem ser classificados como *hard* e *soft*, sendo assim para gerar uma tabela viável é necessário satisfazer todas as restrições do tipo *hard* e preferencialmente, porém não obrigatoriamente, satisfazer as restrições do tipo *soft*.

Conforme Austregésilo (2014 apud SCHAEFER, 1999) nos problemas de *timetabling*, existem três classificações diferentes, estas são:

- i. *Schooltimetabling*: problemas de programação semanal de horários entre professor e turma nas instituições de ensino, onde as disciplinas são fixas para cada turma e sendo assim o objetivo é não permitir que um mesmo professor fique alocado em mais de uma turma ao mesmo tempo.
- ii. *Coursetimetabling*: problemas de programação semanal de todas as disciplinas para todos os períodos dos cursos universitários, o que termina a relação entre docente/turma/horário, a diferença do *schooltimetabling* é que os estudantes podem ter a opção de escolha de matérias que deseja cursar.
- iii. *Examinationtimetabling*: problemas de programação de exames/provas, para cursos universitários, tendo como objetivos evitar colisões de horários de provas onde a mesma turma tenha 2 ou mais disciplinas diferentes e distanciar as datas das provas o máximo possível.

Problemas de *timetabling* são comuns em todas as instituições de ensino, pois para distribuir os horários de aula é necessário atender a diversas restrições, seja de turno, professores ou turmas. Geralmente as principais restrições são os conflito de aulas; disciplinas pertencentes ao mesmo semestre do mesmo curso não podem se encontrar no mesmo horário; local mais apropriado para a alocação da disciplina (sala ou laboratório, dependendo de recursos e equipamentos solicitados para cada aula); disponibilidade do docente que tem limite de horas de trabalho e o mesmo pode possuir mais de uma disciplina em mais de um curso, sendo assim ele não pode ministrar aulas de turmas diferentes no mesmo horário. Essas restrições são consideradas as de maior peso, pois não pode infringi-las.

Uma solução para este problema é encontrar uma grade horária satisfatória que atenda todas as restrições de maior peso e as de menor peso na sua maioria, satisfazendo assim a necessidade da instituição. Alguns métodos auxiliam na resolução do problema de *timetabling*, que são os que utilizam os modelos de árvores de busca, e de heurísticas para limitar a busca, sem essa limitação seria impossível alcançar todo o espaço de busca, devido

ao mesmo ser muito grande, é quase impossível assegurar que será obtido o mesmo resultado final e uma solução ótima.

2.2. O PROBLEMA DA GRADE DE HORÁRIOS

Para que se entenda o problema, é necessário entender o funcionamento de uma grade horária em uma instituição de ensino. Uma grade horária nada mais é do que uma tabela, onde docentes e alunos verificam dias e horários de cada disciplina e curso. Os usuários principais da grade horária são:

- O professor no qual a grade horária funciona como uma agenda de horários que determina o horário em que cada disciplina de cada turma e curso será ministrada.
- O aluno no qual sua agenda escolar estará guiada pelos horários das disciplinas que ele irá cursar.
- A instituição de ensino, que distribui seus recursos humanos e materiais (como salas, laboratórios e equipamentos).

O estudo será realizado com base na estrutura de uma Instituição de Ensino Superior que é formado por 7 cursos distribuídos em 3 períodos (matutino, vespertino e noturno), onde um mesmo curso pode ter turmas em mais de um período. Cada curso possui em média 36 disciplinas distribuídas em 6 semestres. As aulas do semestre são distribuídas semanalmente, o período matutino e vespertino possuem 6 horários de aulas cada, disponíveis para alocação da mesma, o período Noturno possui 4 horários de aula disponíveis de segunda à sexta e 6 horários disponíveis no sábado. As aulas dos períodos matutino e vespertino são distribuídas entre 4 a 6 aulas dependendo do curso, as aulas possuem 50 minutos cada e são agrupadas de duas em duas, sendo assim um total de uma hora e quarenta minutos.

Para ter um resultado satisfatório é necessário atender a diversas restrições, sendo que alguma delas não podem ser infringidas, essas são chamadas de restrições rígidas, já outras não irão impactar de forma negativa

no resultado final, essas são chamadas de restrições flexíveis. As restrições rígidas podem ser definidas como:

- i. Uma turma do mesmo semestre pertencente ao mesmo curso não pode ter mais de uma disciplina no mesmo horário.
- ii. Disponibilidade dos docentes
- iii. Carga horária máxima de 8 aulas por dia
- iv. Interjornada, onde o docente precisa ter 11 horas de descanso entre um dia para o outro.
- v. Um professor não pode lecionar em mais de uma disciplina no mesmo no horário

E uma da restrição flexível levanta é: aulas duplas no mesmo dia da semana.

3. ALGORITMO GENÉTICO

Na década de 40 os pesquisadores se inspiraram na natureza para dar início ao ramo da Inteligência Artificial, a partir daí se iniciou a tentativa de compreensão de processos naturais, raciocínio, aprendizado e lógica (LINDEN, 2012). Em Holland em 1975 propôs um modelo heurístico computacional no qual implementado ofereceria soluções satisfatórias a problemas muito complexos para época. Holland (1975) apresentou os AGs como uma metáfora, do processo de adaptação e evolução no mundo real os simulando computacionalmente, porém os AGs se tornaram uma importante ferramenta da inteligência artificial, sendo assim muito disseminado por cientistas da computação.

“Algoritmos genéticos são um ramo dos algoritmos evolucionários e como tal podem ser definidos como uma técnica de busca baseada numa metáfora do processo biológico de evolução natural” (LINDEN, 2012, p. 46)

A seleção natural é um processo, onde uma população possui diversos indivíduos, e os indivíduos mais aptos são os melhores a se reproduzir e gerarem filhos que farão parte de uma nova população com indivíduos cada vez melhores. Isto para o meio computacional faz com que seja possível criar um AG para que atenda um problema no qual é necessário encontrar uma melhor solução entre muitas.

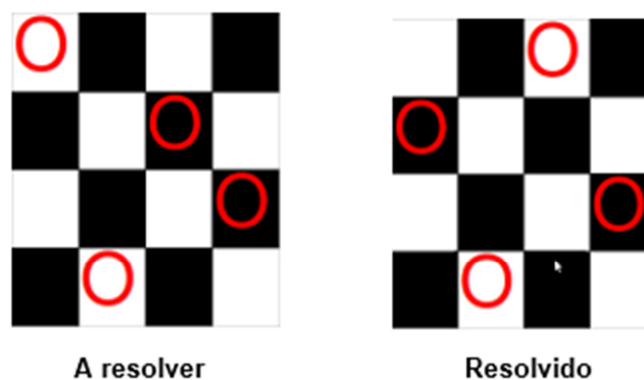
Os AG são algoritmos de otimização global, onde se emprega uma estrutura de busca estruturada e paralela, porém aleatória, voltada na direção de pontos de alta aptidão, em outras palavras, ele encontra pontos que correspondam melhor a solução objetivada, para isso são realizadas diversas iterações, que são conhecidas como gerações. Para poder se utilizar os AG como solução para um problema deve se ter como início a representação do problema de maneira que o AG possa agir satisfatoriamente sobre ele (RAMOS, 2002).

Segundo Linden (2012) a parte mais importante de um AG é a codificação do cromossomo, assim como a função de avaliação, esta codificação cromossômica deve ser realizada de modo inteligente, ou seja, ser

bem formulada, já conter as restrições e o espaço de busca bem definidos. A reprodução e mutação são feitas em indivíduos selecionados dentro da população, esta seleção deve ser feita com mais frequência em indivíduos mais aptos, para que assim possam predominar, ficando com uma população com melhores indivíduos.

Nakashima (2009) exemplifica de forma simples o problema da N-Rainhas. No xadrez a Rainha é a peça mais poderosa, que pode se locomover em qualquer direção, e em quantas casas quiser. Em um jogo de xadrez existem apenas duas rainhas, porém no exemplo do N-Rainhas se possui n rainhas e um tabuleiro com n linhas e n colunas, o objetivo é posicionar as n rainhas no tabuleiro de forma que nenhuma delas esteja se atacando. Por exemplo: um tabuleiro com 4 colunas e 4 linhas, tendo como desafio posicionar 4 rainhas de modo que elas não estejam se atacando. Este desafio é possível resolver com facilidade, até mesmo visualmente, como exibido na Figura 1.

Figura 1 - Exemplo N-Rainhas



Fonte: Nakashima (2009).

Porém se necessário resolver o mesmo problema com o valor de n muito elevado, já não é tão simples resolvê-lo, pois seria necessário um número muito extenso de verificações e poderiam surgir diversas soluções para o mesmo problema. Logo quando há um problema com um espaço de busca muito extenso para que se possa solucioná-lo, os meios mais comuns e simples são por métodos de busca heurística como os AGs.

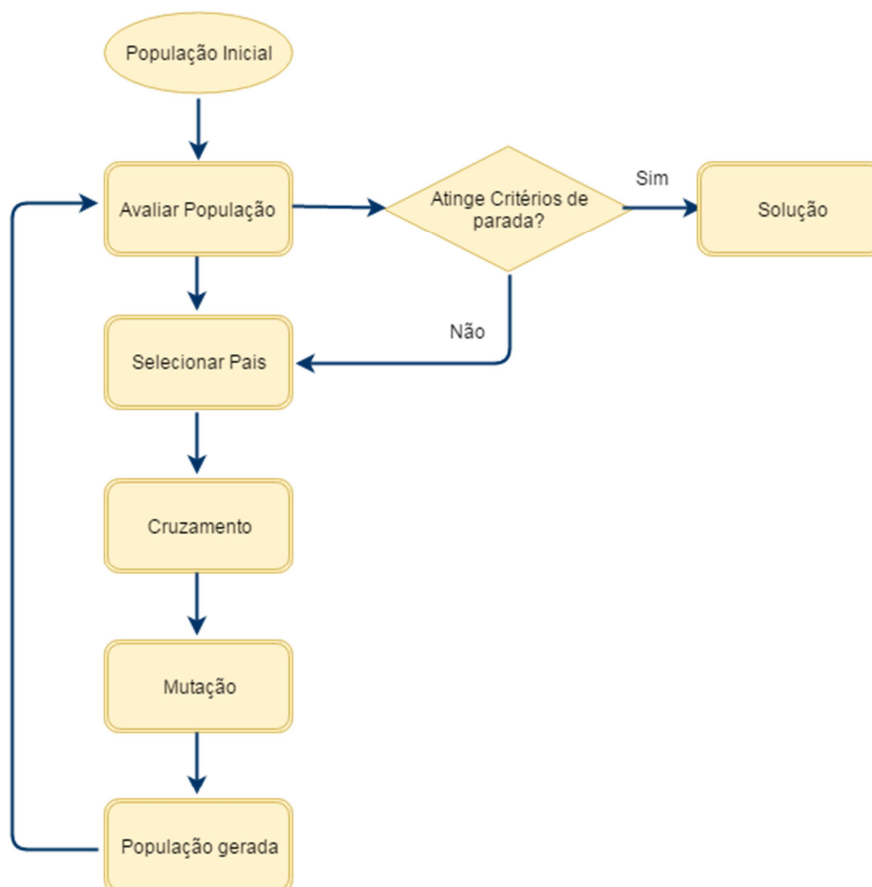
Os AGs são ferramentas onde o resultado final é probabilístico e não determinístico, sendo assim um problema com a mesma população inicial pode chegar a n soluções diferentes, o que o torna quase que impossível de se repetir o mesmo resultado. Os AGs possuem componentes aleatórios, mas como ele utiliza a informação da população corrente para gerar uma próxima população não se pode considerá-lo totalmente aleatório.

No subcapítulo seguinte é demonstrado o funcionamento de um AG com codificação binária para que se possa compreender como ele consegue solucionar da melhor maneira um problema que pode conter diversas soluções.

3.1. FUNCIONAMENTO GENÉRICO DO ALGORITMO

O funcionamento genérico dele se dá por: i) gerar uma população de cromossomos; ii) avaliar cada cromossomo na população; iii) selecione os pais de acordo com sua adequação (melhor adequação, mais chances de ser selecionado) para gerar novos cromossomos (filhos); iv) utilizar os operadores para realizar uma recombinação e mutação aos pais de forma a gerar novos indivíduos (cromossomos) para a próxima geração; v) Gerar uma nova população; vi) se atingir o critério de parada, retorne o melhor cromossomo, senão, voltar ao passo ii. A execução deste algoritmo é representada de modo simples no fluxograma apresentado na Figura 2.

Figura 2 - Representação do Algoritmo Genético



Fonte: Adaptado de Linden (2012).

Para melhor compreender cada processo e suas características os próximos subcapítulos explicam o funcionamento do mesmo.

3.1.1. REPRESENTAÇÃO GENÉRICA DO CROMOSSOMO

Um fundamento básico de um AG é a representação cromossômica que consiste em uma forma de transparecer a informação do problema visando ser tratada pelo computador. O cromossomo é formado por um conjunto de genes, um gene é uma partícula do cromossomo, sendo esse pedaço indivisível. A representação de um cromossomo segue alguns padrões (LINDEN, 2012), são eles:

- i. A representação do cromossomo deve ser feita de maneira mais simples o possível.

- ii. Se houver alguma solução restrita ao problema a mesma não deve preferencialmente ser representada.
- iii. Se o problema estabelecer algum tipo de condição, estas condições devem estar implícitas na representação.

Após estruturar a representação do cromossomo pode-se enfim gerar a população inicial.

3.1.2. A POPULAÇÃO INICIAL

A geração da população inicial dos AGs na maioria é gerada de forma aleatoriamente, o que consiste em apenas criar n indivíduos aleatórios e distribuir nos espaços disponíveis. Este tipo de inicialização normalmente é satisfatória, pois gera uma boa distribuição de soluções no espaço de busca. Porém é possível gerar esta população inicial já com algumas diretrizes pré-determinadas para que assim seja mais rápido o processamento das iterações que irão determinar a solução do problema. Após a geração da população podemos então realizar a avaliação dos indivíduos da população.

3.1.3. AVALIAÇÃO DE INDIVÍDUOS

Após a geração da população inicial é necessário que se avalie seus indivíduos para poder selecionar aqueles que irão ser utilizados pelos operadores. Com isso os AGs utilizam uma função de avaliação, esta função irá determinar a qualidade de um indivíduo como possível solução do problema. A função é como se fosse uma nota dada a cada indivíduo e a partir dessa nota será escolhido o indivíduo que será selecionado, sendo essa função um mecanismo capaz de separar os bons e os maus indivíduos para solução.

A função de avaliação na maioria das vezes é a única ligação do programa com o problema a ser resolvido, pois é apenas ela que julga a satisfação da solução apresentada pelo indivíduo. Também chamada de

função de custo, a função de avaliação calcula um valor numérico (o valor do *fitness*) que define a que ponto os parâmetros apresentados satisfazem o problema, ou seja, utiliza todos os valores armazenados no cromossomo e retorna um valor numérico, este valor é uma métrica de qualidade da solução. AGs são técnicas de otimização na qual avalia se o cromossomo c_1 é melhor que o cromossomo c_2 , caso o valor de *fitness* de c_1 deve ser maior (problemas de maximização) ou menor (problemas de minimização) em relação a c_2

3.1.4. ELITISMO

Elitismo é uma pequena alteração na população, que garante que a cada geração o AG tenha uma melhor taxa de desempenho. Tendo em vista que, na maioria dos casos a avaliação mede a adequação do indivíduo a solução, os indivíduos mais aptos são salvos e inseridos na próxima geração. Isso garante que mantemos na população os esquemas responsáveis pelas boas avaliações das melhores soluções e conseqüentemente aumenta sua capacidade de aproveitamento, porém sem danos significativos ao seu desempenho.

3.1.5. MÉTODOS DE SELEÇÃO

O fundamento da seleção de pais é que devesse privilegiar os indivíduos que obtenham uma nota de avaliação melhor, ou seja, aqueles que melhor atendem a solução do problema devem ser preservados ao mesmo tempo que não se deve desprezar logo de cara os indivíduos que possuem uma nota excessivamente ruim. Este tipo de decisão pois um indivíduo mesmo com uma péssima avaliação pode ter características genéticas favoráveis a criação de um novo indivíduo que seja bom para a solução do problema, características essas que podem não estar presentes em nenhum outro indivíduo.

Outro fator é que, se manter apenas os melhores indivíduos a população tenderá a ser composta apenas por indivíduos semelhantes, o que acarretará

na falta de diversidade populacional, dificultando assim a evolução da mesma. Esse efeito é chamado de convergência genética que pode ser resolvido selecionando de forma justa os indivíduos menos aptos esse tipo de problema é evitado.

Um método amplamente utilizado é a roleta:

“A maneira que a grande maioria dos pesquisadores de GA* utiliza é o método da roleta viciada. Neste método criamos uma roleta (virtual) na qual cada cromossomo recebe um pedaço proporcional à sua avaliação (a soma dos pedaços não pode superar 100%). Depois rodamos a roleta e o selecionado será o indivíduo sobre qual ela parar.” (LINDEN, 2012, pg. 77)

Nota-se que neste método a roleta é girada aleatoriamente n vezes, e cada indivíduo recebe um “pedaço” da roleta, sendo assim os melhores indivíduos tem mais chances de serem selecionados mais vezes, pois é dado um pedaço maior a ele, enquanto um indivíduo menos apto recebe um pedaço menor da roleta, porém mesmo com essa desvantagem ainda assim ele tem chances de ser selecionado para a reprodução.

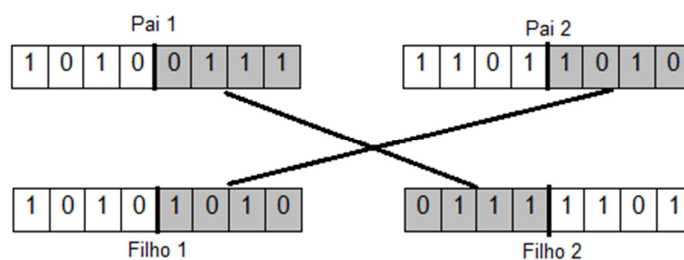
3.1.6. OPERADOR DE *CROSSOVER*

Existem diversos tipos de operadores que realizam uma recombinação cromossômica (*crossover*), porém os mais comuns são: de um ponto, de dois pontos e uniforme.

O *crossover* de um ponto é um dos mais simples, e funciona da seguinte maneira: Após selecionar dois pais, é selecionado um ponto de corte, que é uma posição entre dois genes de um cromossomo, este ponto de corte é o ponto onde será separado cada um dos genes que compõe o pai. Após sortear o ponto de corte, o pai é separado em duas partes e não é necessário que as partes tenham o mesmo tamanho. O primeiro filho será formado pela concatenação da primeira parte do pai 1, e a segunda parte do pai 2. Já o segundo filho será formado com o restante, sendo o que resta do pai 1 é a

segunda parte do cromossomo e com o que resta do pai 2, a primeira parte do cromossomo antes do ponto de corte. Este processo é similar com o que ocorre na natureza, na reprodução sexuada, sendo a única diferença que na natureza não há apenas um ponto de corte. Este processo é realizado como no exemplo da figura 3:

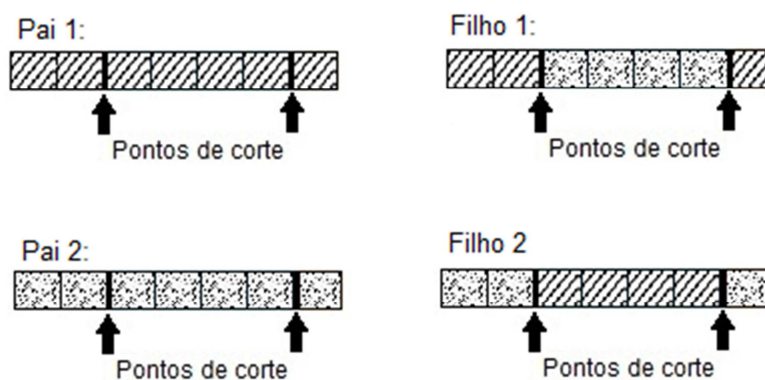
Figura 3 - *Crossover* um ponto de corte



Fonte: Linden (2012)

Já o *crossover* de dois pontos de corte, possui um funcionamento semelhante ao de um ponto, tendo como diferença que serão sorteados aleatoriamente dois pontos de corte. Para exemplificar seu funcionamento abaixo na Figura 4.

Figura 4 - *Crossover* dois pontos de corte



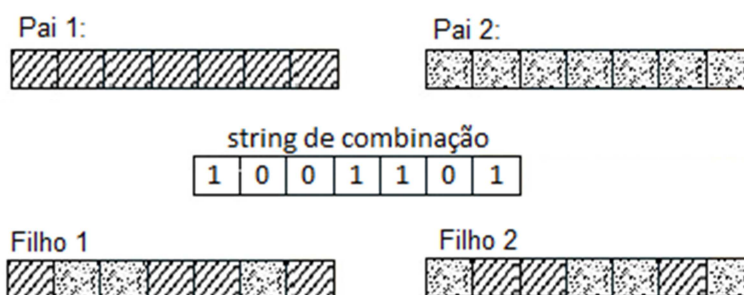
Fonte: Linden (2012)

Note que na Figura 4 o primeiro filho é composto pela primeira parte antes do ponto de corte do primeiro pai, a parte do meio é composta pela parte dentro dos pontos de corte do segundo pai e a última parte é composta pela

parte após o segundo ponto corte do primeiro pai. Já o segundo filho é composto pelo material genético restante.

Outro *crossover* comum é o *crossover* uniforme que funciona da seguinte forma: para cada gene do pai é sorteado um número entre 0 ou 1, o primeiro filho será composto pelos genes do primeiro pai onde o valor for igual a 1 e onde o valor for 0 serão atribuídos os genes do segundo pai. Para melhor compreensão a Figura 5 exibe um exemplo de seu funcionamento.

Figura 5 - *Crossover* unilateral



Fonte: Linden (2012)

Como pode-se observar na Figura 5, foi sorteada uma *string* que possui o mesmo tamanho do pai, é então verificado onde para cada posição de gene do pai, na mesma posição o valor na *string* é igual a 1, caso o valor seja 0 o filho será composto pela mesma posição da *string* da posição do segundo pai.

3.1.7. OPERADOR DE MUTAÇÃO

O processo de mutação pode ser realizado de forma conjunta com os operadores escolhidos, como por exemplo pode agir em conjunto com o operador de *crossover*. Após os filhos estarem formados pelo processo dos operadores de *crossover*, se inicia o processo de mutação, que funciona da seguinte maneira: este operador de mutação tem uma probabilidade muito baixa (da ordem 0,5%) associada, onde é sorteado um número entre 0 e 1, se este número for menor que a probabilidade predeterminada então o gene sofre

mutação, alterando o valor do gene em questão aleatoriamente, este processo é repetido em todos os genes que compõe o filho.

Esta probabilidade que decide se o gene sofrerá ou não a mutação é uma das propriedades do AG onde apenas a experiência pode determinar, o fundamento para se obter este valor da probabilidade é que ele deve ser baixo, pois se fosse constituído de valores muito alto a qualidade população logo iria se degenerar e dificilmente se obteria uma boa solução. Diversos textos sugerem que este operador não atue de forma aleatória e que aja apenas quando selecionado, e que altere o conteúdo do gene apenas para outro conteúdo válido geneticamente.

3.1.8. NOVA POPULAÇÃO

Assumindo que a população não pode crescer, pode-se então admitir que os pais tenham que ser substituídos pelos filhos que vem sendo gerados. Com a nova população gerada a mesma será avaliada novamente, para que todo o processo se repita até que se atinja o critério de parada ou o número máximo de iterações.

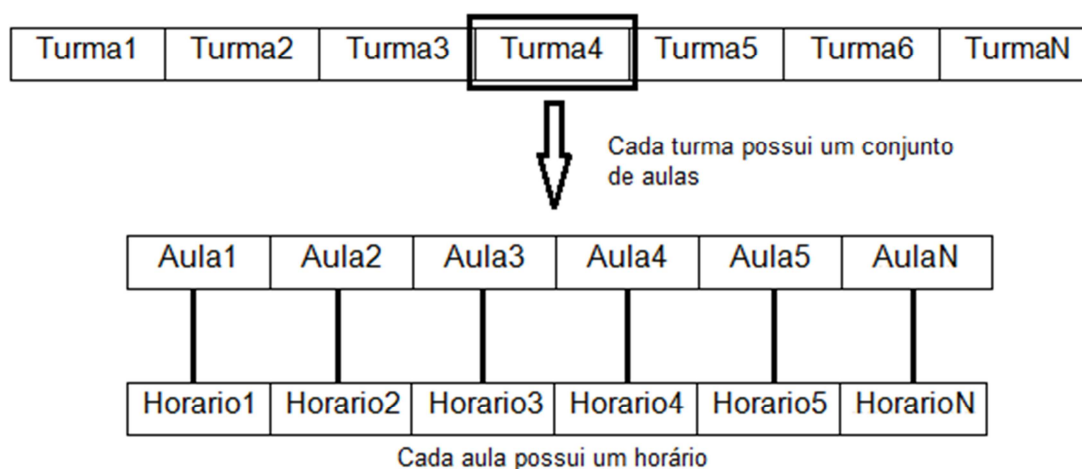
4. ALGORITMO GENÉTICO PROPOSTO

De acordo com o problema de *timetabling* aplicado uma Instituição de Ensino Superior, é proposto então um AG para que assim possa ser gerada automaticamente uma grade horária satisfatória. Os detalhes desse AG proposto são apresentados nos próximos subcapítulos.

4.1. O CROMOSSOMO

O cromossomo é composto por uma lista de aulas (cada aula possui, turma, horário e atribuição) e cada aula irá receber um horário aleatório, dentro dos horários possíveis para uma turma. A Figura 6 apresenta essa estrutura.

Figura 6 - Estrutura do Cromossomo proposto



Fonte: Próprio Autor

Ao adicionar o horário, é verificado se ele pode ser atribuído em relação ao período, ou seja, um horário das 07:40hrs, só poderá ser atribuído a aula em que o período da turma esteja no período matutino. Isso contribui para geração de um cromossomo factível para solucionar o problema.

4.2. POPULAÇÃO INICIAL

Para que se obtenha uma população inicial foi definido o tamanho da população (quantidade de cromossomos) e a quantidade de épocas (iterações) a qual a população irá passar. A população como apresentada no Capítulo 3 é um conjunto de cromossomos (indivíduos) gerados aleatoriamente.

4.3. AVALIAÇÃO

Para a avaliação, cada indivíduo (cromossomo) é avaliado de acordo com as restrições estabelecidas no Quadro 1, caso ocorra infração de restrições o indivíduo será punido.

Quadro 1. Restrições e notas

Restrição	Tipo	Nota atribuída
Caso o professor estiver indisponível no horário atribuído	Rígida	100
Professor não pode ministrar aulas no mesmo horário	Rígida	50
A turma não pode ter aula no mesmo horário (esta restrição já foi atendida pela estruturação do cromossomo)	Rígida	10
Horários devem estar dentro do período de horas (esta restrição está atendida tendo em vista a estrutura do cromossomo, já que só serão atribuídos horários de um mesmo período).	Rígida	40
Professor não pode ter 8 aulas seguidas	Rígida	80
Aulas que possuem 4 créditos deverão preferencialmente serem ministradas no mesmo dia sem intervalo	Flexível	10

Fonte: Próprio Autor

O *fitness* é o que determina a qualidade da solução, quanto maior o *fitness* pior a solução neste caso, pois a solução perfeita seria se o valor do *fitness* fosse igual a 0. Caso contrário, significa que o indivíduo recebeu algumas punições.

4.4. ELITISMO

A cada geração são armazenados 10 indivíduos que possuem um melhor valor de *fitness* (valor mais baixo), esses indivíduos serão adicionados a próxima população, isso preserva os melhores indivíduos, o que faz com que a população sempre evolua e obtenha cada vez melhores indivíduos.

4.5. MÉTODO DE SELEÇÃO ROLETA

A roleta funciona da seguinte forma: é sorteado um número aleatório, de acordo com a avaliação de um determinado indivíduo, o mesmo recebe um pedaço da roleta, indivíduos com notas menores (*fitness*), ou seja, indivíduos mais aptos recebem um pedaço maior e proporcional com sua avaliação, sendo assim um indivíduo mais apto tem mais chances de ser selecionado, porém um pior indivíduo ainda assim possui uma mínima chance de ser selecionado. A equação (1) apresenta o cálculo da probabilidade de seleção do indivíduo (área da roleta).

$$p_i = \frac{\max(c) - f(c_j)}{\sum_j^n f(c_j)} \quad (1)$$

onde p_i é a probabilidade do indivíduo ser selecionado; $f(c_j)$ é o *fitness* do cromossomo (avaliação); c_j é o cromossomo j ; ou seja, a probabilidade p_i de seleção de um indivíduo é igual ao valor mais alto de avaliação dentre os

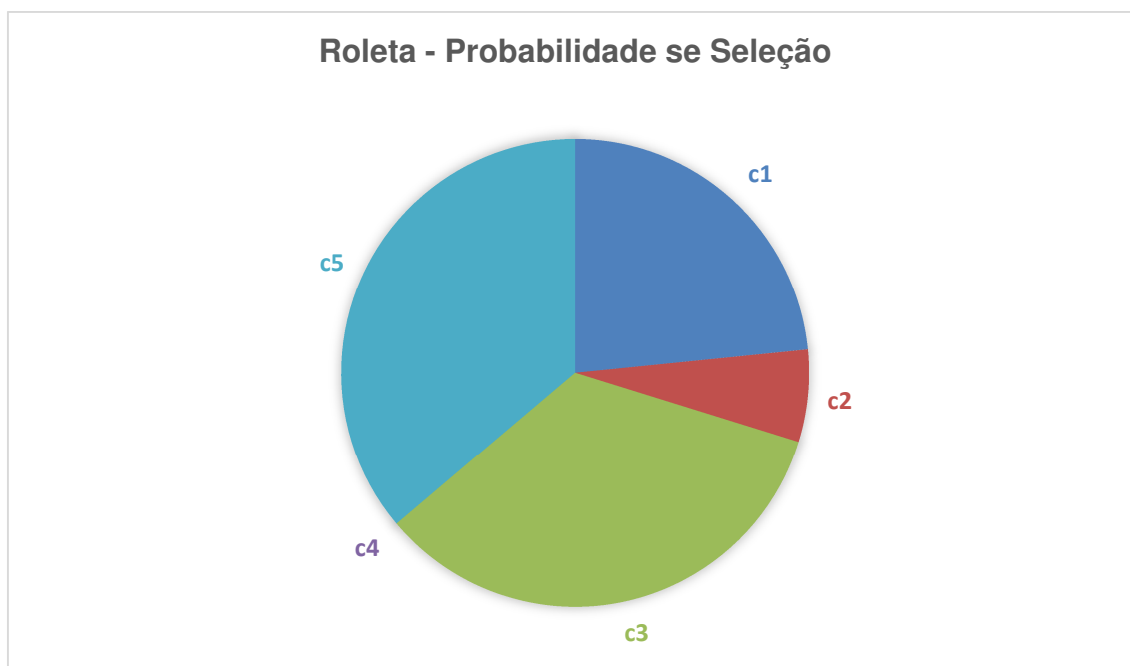
cromossomos, menos a avaliação do cromossomo c_j , dividido pelo total de *fitness* de todos os indivíduos. A tabela 1 mostra um exemplo da roleta.

Tabela 1 - Exemplo de dados para o método da roleta

c_j	$f(c_j)$	p_j (%)
C1	80	22,91
C2	160	6,25
C3	30	33,33
C4	190	0
C5	20	35,41
Total	480	98,90

Fonte: Próprio Autor

Gráfico 1 - Roleta de Probabilidade de Seleção



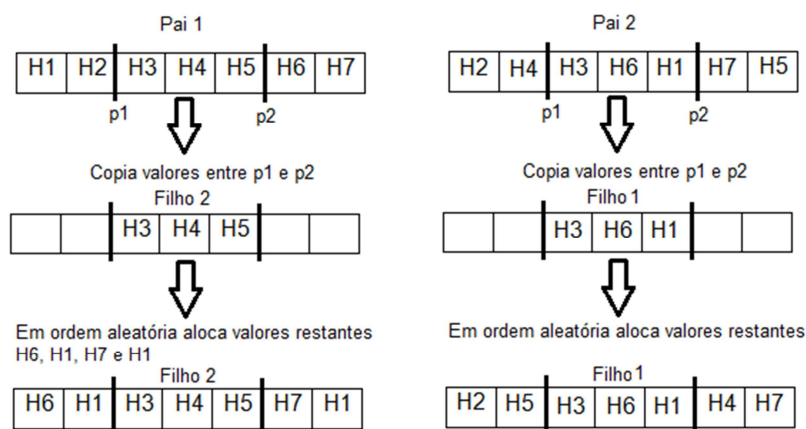
Fonte: Próprio Autor

4.6. CRUZAMENTO (*CROSSOVER*)

O método de cruzamento utilizado para solução do problema proposto é o *crossover* por ordem, pois os horários das aulas estão agrupados por turma, e o mesmo só pertence a turma se o horário pertencer ao período. Os horários serão permutados de forma que não se repitam na mesma turma.

O *crossover* por ordem funciona da seguinte forma: são dados dois pontos de corte, em seguida os genes do indivíduo são alocados temporariamente em uma lista até o primeiro ponto de corte, do primeiro ponto de corte até o segundo ponto de corte é copiado o conteúdo para o filho 1, do segundo ponto de corte em diante o conteúdo é inserido na lista temporária, a lista temporária é embaralhada seus valores, e logo em seguida os valores da lista preencherão o restante do filho 1. O mesmo procedimento ocorre com o pai 2 e filho 2, como no exemplo da Figura 7.

Figura 7 - *Crossover* por ordem exemplo



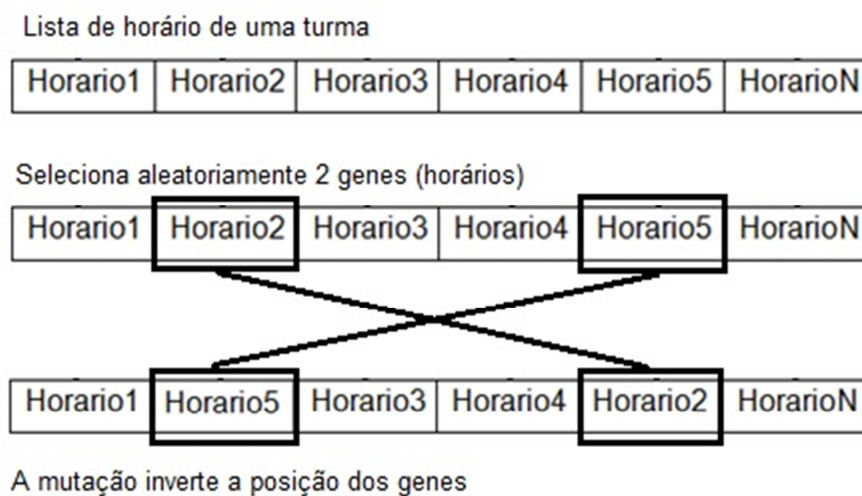
Fonte: Próprio Autor, baseado em Linden(2012)

4.7. MUTAÇÃO

A utilização da mutação no trabalho proposto é realizando apenas a permutação de dois genes, dois horários aleatórios trocam de posição entre si, como é possível observar na figura 8. Para que não existam resultados

infectíveis apenas horários da mesma turma são permutados. Os genes que irão ser multados são definidos aleatoriamente.

Figura 8 - Exemplo de Mutação



Fonte: Próprio autor

4.8. CRITÉRIO DE PARADA

Após o cruzamento e mutação a população passa por uma nova avaliação, e logo após avaliar, a geração é salva, caso atinja o critério de parada (número de iterações/quantidade de épocas) é verificado o melhor indivíduo (aquele que possuir o menor *fitness*), o melhor indivíduo é a solução da grade horária.

5. IMPLEMENTAÇÃO DO SISTEMA

Este capítulo apresenta as ferramentas utilizadas para desenvolvimento do sistema e a implementação do mesmo.

5.1. FERRAMENTAS UTILIZADAS

Este trabalho utiliza o banco de dados MySQL 5.2 (versão gratuita). Este sistema gerenciador de banco de dados (SGBD) é disponibilizado pela empresa MySql, junto com as demais ferramentas, plug-ins, drivers e afins. O banco de dados foi criado utilizando a IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento é um software que reúne características e ferramentas de apoio ao desenvolvimento) MySQL Workbench que tem diversos recursos para modelagem, criação e gerenciamento do mesmo.

Para codificação foi utilizado a linguagem Java que é uma linguagem orientada a objetos juntamente com a IDE NetBeans na sua versão 8. Para a comunicação do projeto com o banco de dados foi utilizado o Hibernate que é um framework de persistência. Este framework funciona como um facilitador de acesso ao banco de dados, assim como na recuperação de dados, na inserção, alteração e na exclusão de dados.

Para o desenvolvimento do diagrama de classes foi utilizado o Astah Community na sua versão 7.1, sendo a versão Profissional com a licença de estudante.

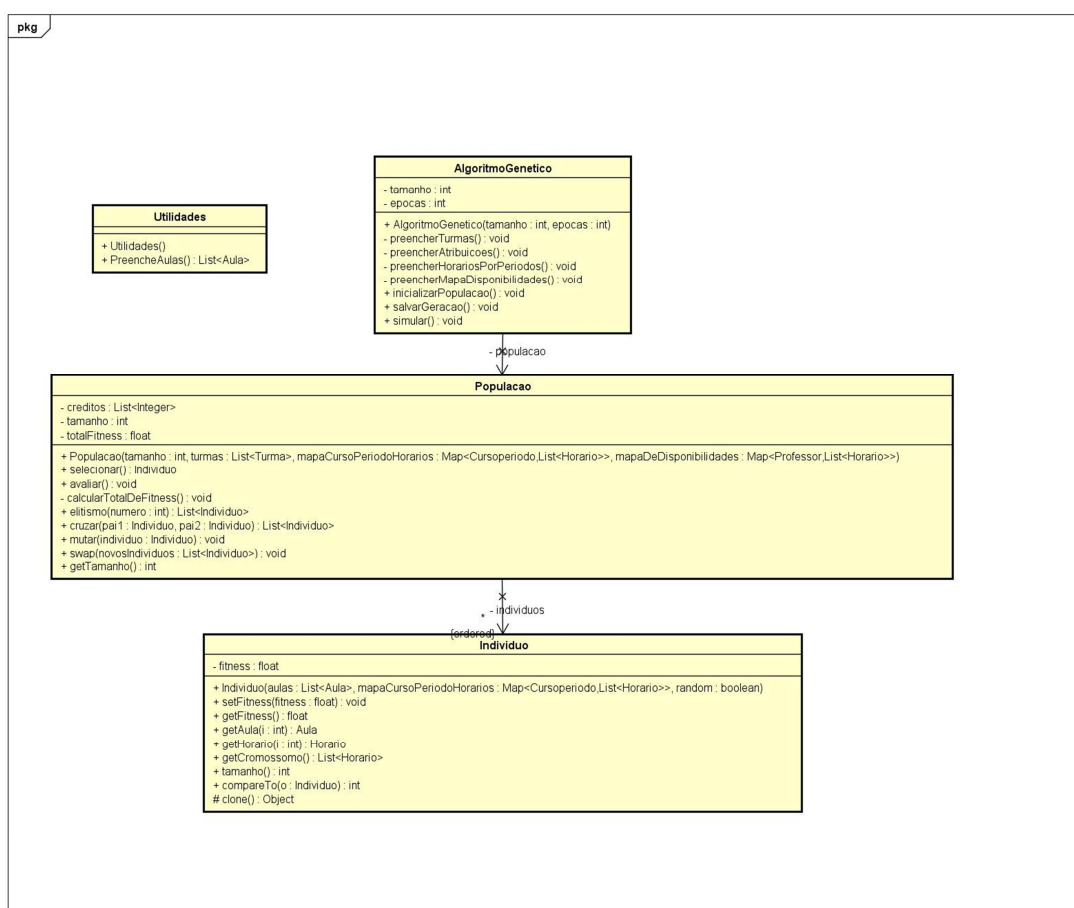
5.2. DIAGRAMA DE CLASSES

Segundo Guedes (2009) “Diagramas de classes representam as estruturas de classes utilizadas pelo sistema, apresentando seus atributos e métodos, além de demonstrar seus relacionamentos entre as classes. ”

Diagramas são representações gráficas de alto nível de como o sistema irá se comportar. No Diagrama de Classes são demonstradas as classes de um sistema, seus atributos e métodos desenvolvidos no projeto, graficamente as classes são representadas por retângulos incluindo nome da classe, seus atributos e métodos.

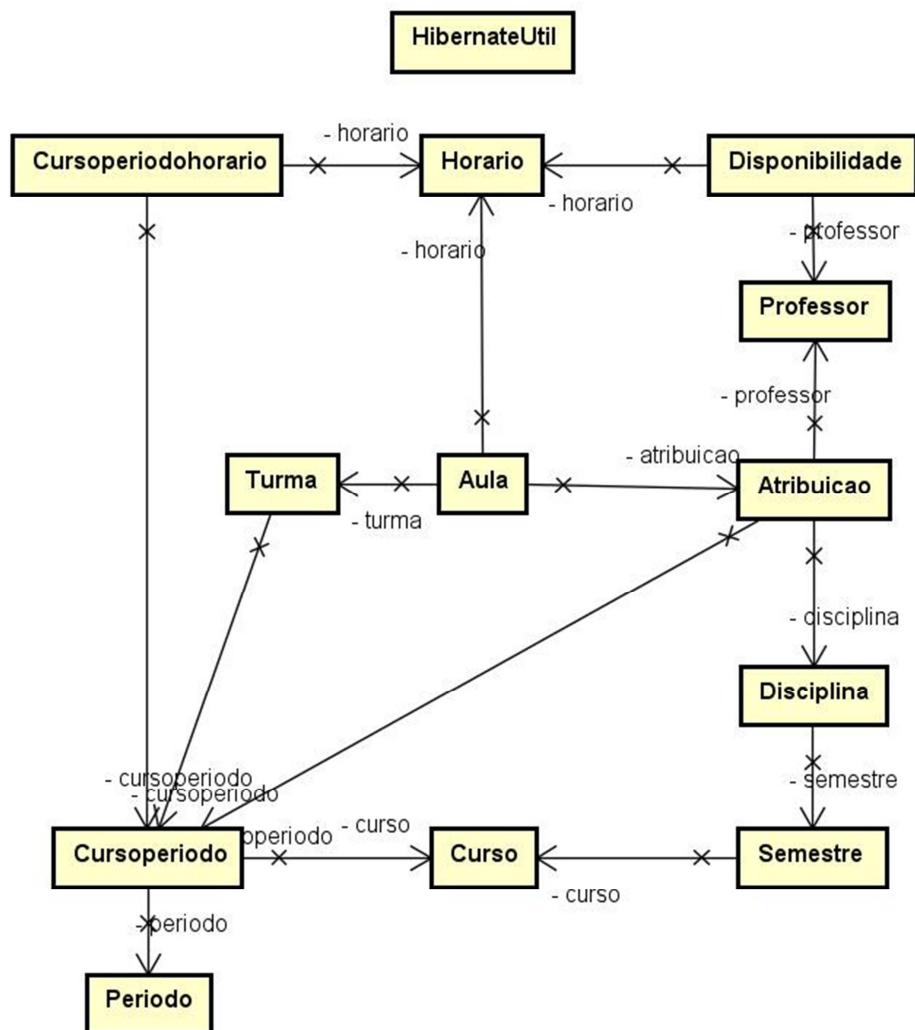
Para este trabalho foram desenvolvidas as classes (Algoritmo Genético, Indivíduo e População) e a estrutura do banco de dados, as classes referentes as tabelas no banco de dados foram geradas automaticamente pelo Hibernate (classes: Atribuicao, Aula, Curso, CursoPeriodo, CursoPeriodoHorario, Disciplina, Disponibilidade, Horario, Periodo, Professor, Semestre e Turma). A Figura 9 e 10 apresentam esses diagramas.

Figura 9 - Diagrama de Classes Algoritmo Genético



Fonte: Próprio Autor

Figura 10 - Diagrama de Classes – Classes de Persistência



Fonte: Próprio Autor

5.3. DIAGRAMA DE ENTIDADE RELACIONAMENTO

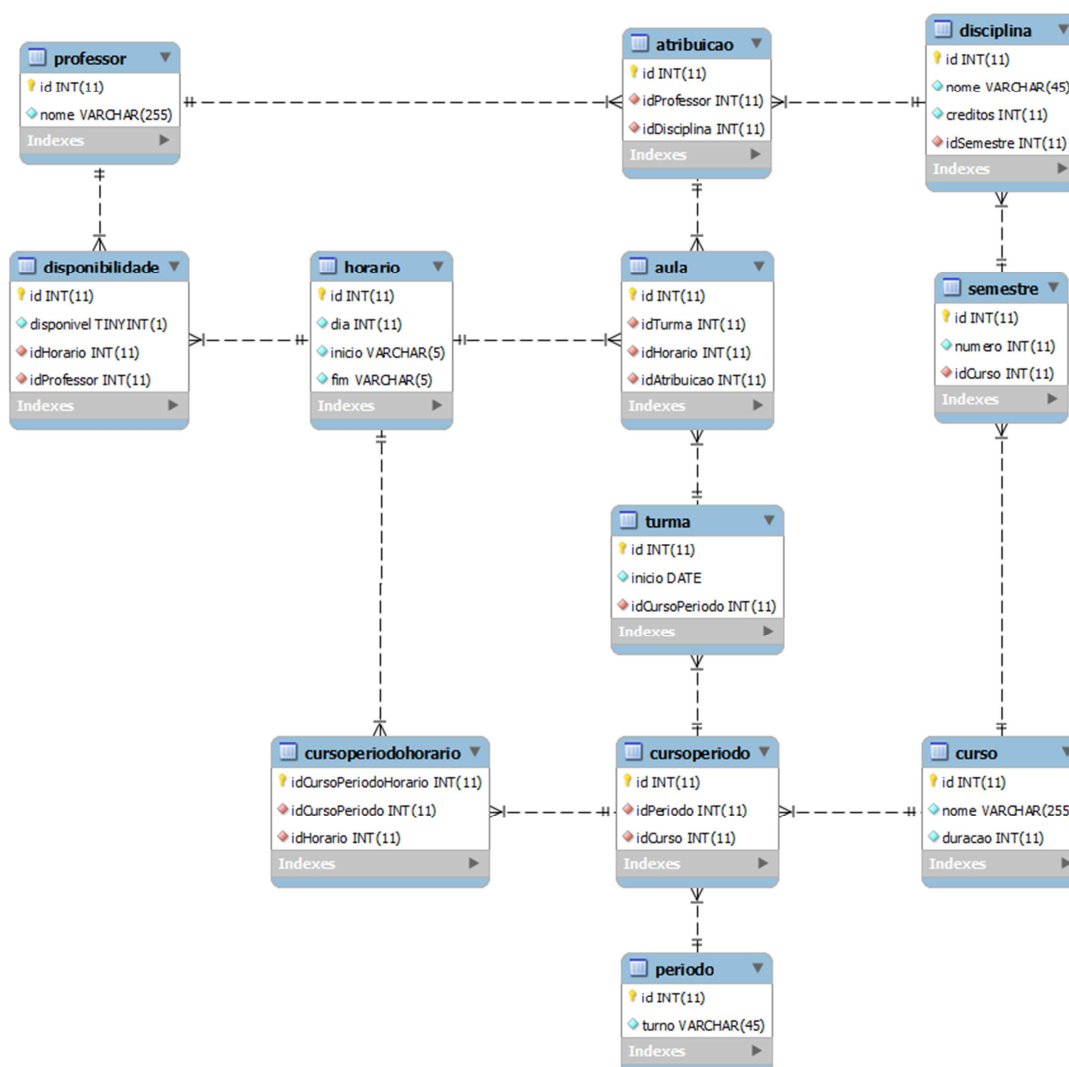
Para se ter um banco de dados de qualidade é necessário realizar a modelagem de dados, que nada mais é do que a análise e determinação de forma conceitual como estarão dispostos os dados, como os mesmos irão se relacionar.

A representação conceitual de um banco de dados pode ser realizada com um diagrama de entidade relacionamento, que é uma representação gráfica, onde são descritos os dados ou aspectos de um banco de dados. Os

principais elementos dos modelos de entidades-relacionamento (MER) são as próprias entidades (objetos, tabelas), suas relações com as demais, seus atributos e tipos de atributos.

Para representar as tabelas existentes no banco de dados (BD) de forma a facilitar a visualização do relacionamento entre elas além de seus atributos, o diagrama de entidade relacionamento realiza essa demonstração, este diagrama da figura 11 foi gerado automaticamente pelo MySQL Workbench a partir do banco de dados criado para o projeto.

Figura 11 - Diagrama Entidade Relacionamento



Fonte: Próprio Autor

A partir da elaboração do diagrama entidade relacionamento é possível extrair um dicionário de dados como se pode verificar nos quadros a seguir, para que fique mais simples a compreensão do banco de dados.

Um dicionário de dados é um conjunto de informações sobre os metadados (dados sobre os dados) que contém as definições dentro do contexto do banco de dados.

Quadro 1 - Tabela BD - Atribuição

Atribuição – Relaciona a qual Disciplina um professor pertence		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
idProfessor	INT(11)	Identificador de Professor
idDisciplina	INT(11)	Identificador de Disciplina

Fonte: Próprio Autor

Quadro 2 - Tabela BD - Aula

Aula – Relaciona a qual turma, horário e Atribuição uma aula possui		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
idTurma	INT(11)	Identificador da Turma
idHorario	INT(11)	Identificador de Horário
idAtribuicao	INT(11)	Identificador de Atribuição

Fonte: Próprio Autor

Quadro 3 - Tabela BD - Curso

Curso		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
nome	VARCHAR(255)	Nome do Curso

duracao	INT(11)	Duração em horas
---------	---------	------------------

Fonte: Próprio Autor

Quadro 4 - Tabela BD - CursoPeriodo

CursoPeriodo – Relaciona a qual período um curso pertence		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
idPeriodo	INT(11)	Identificador de Período
idCurso	INT(11)	Identificador de Curso

Fonte: Próprio Autor

Quadro 5 - Tabela BD - CursoPeriodoHorario

CursoPeriodoHorário – Relaciona a qual Curso por Período um horário pertence		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
idCursoPeriodo	INT(11)	Identificador de CursoPeriodo
idHorario	INT(11)	Identificador de Horário

Fonte: Próprio Autor

Quadro 6 - Tabela BD - Disciplina

Disciplina – Contem a disciplina e a relaciona a um Semestre (que contém Curso)		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
nome	VARCHAR(45)	Nome da Disciplina
creditos	INT(11)	Quantidade de créditos (2 ou 4 aulas)
idSemestre	INT(11)	Identificador do Semestre

Fonte: Próprio Autor

Quadro 7 - Tabela BD - Disponibilidade

Disponibilidade – Relaciona professor a horário onde o mesmo esteja disponível		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
disponivel	TINYINT(1)	Se está disponível = 1, senão 0
idHorario	INT(11)	Identificador de Horário
idProfessor	INT(11)	Identificador de Professor

Fonte: Próprio Autor

Quadro 8 - Tabela BD - Horario

Horario – Contem horários e dia da semana		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
dia	INT(11)	Dia da semana de 1 a 6
inicio	VARCHAR(5)	Horário de Início da aula
fim	VARCHAR(5)	Horário de fim da aula

Fonte: Próprio Autor

Quadro 9 - Tabela BD - Periodo

Periodo – Contém o nome do período		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
turno	VARCHAR(45)	Período (matutino, vespertino e noturno)

Fonte: Próprio Autor

Quadro 10 - Tabela BD - Professor

Professor – Contém o nome do professor		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela

nome	VARCHAR(255)	Nome do Professor
------	--------------	-------------------

Fonte: Próprio Autor

Quadro 11 - Tabela BD - Semestre

Semestre – Relaciona um semestre com um curso		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
numero	INT(11)	Número do semestre (de 1 a 6)
idCurso	INT(11)	Identificador do Curso

Fonte: Próprio Autor

Quadro 12 - Tabela BD - Turma

Turma – Relaciona uma turma a um curso por período		
Campo	Tipo	Descrição
id	INT(11)	Identificador da tabela
inicio	DATE	Data de inicio de uma turma
idCursoPeriodo	INT(11)	Identificador do CursoPeriodo

Fonte: Próprio Autor

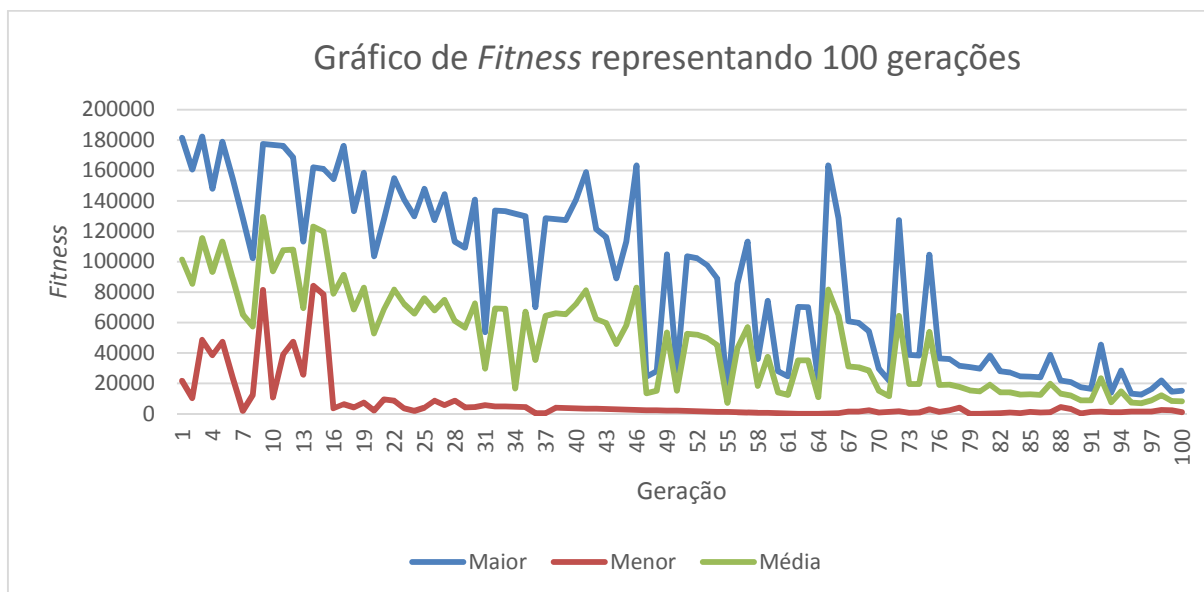
6. EXPERIMENTOS E RESULTADOS

Para os testes foram cadastradas todas as informações referentes a distribuição da grade horária, tendo uma quantidade de 813 aulas (as aulas foram alocadas dividindo em 2 as disciplinas com 4 créditos) distribuídas e como *fitness* máximo o valor de 195120.

Os experimentos realizados são de acordo com os testes de geração de população. Serão observados o *fitness* de cada geração, e realizado um comparativo com seu maior *fitness*, menor e o *fitness* médio.

No experimento apresentado no gráfico 2, o tamanho da população é de 1000 indivíduos e com a quantidade de 100 gerações (iterações). Este número foi escolhido aleatoriamente.

Gráfico 2 - Simulação AG com 100 gerações



Fonte: Próprio Autor

Pode-se observar que a cada geração a população seguinte fica mais apta, com melhores indivíduos, isso ocorre devido ao elitismo aplicado, que garante que os melhores indivíduos permaneçam para a próxima geração.

Apesar da tendência ser o valor de *fitness* diminuir ele nunca irá tender a 0 no seu valor máximo e por seguinte mediano, devido a sua aleatoriedade no preenchimento da população e as operações de *crossover* e mutação.

O AG, conseguiu encontrar uma solução satisfatória já na sua 19ª geração, mesmo que seu *fitness* não tenha atingido o seu valor mínimo 0, que seria o ideal, ainda sim cumpriu-se o objetivo, tendo em vista a restrição de aulas que possuem 4 créditos, que de preferência deveriam ficar juntas, o que contribui com o incremento da nota *fitness*, porém essa não era uma restrição do tipo rígida, não afetando de forma negativa o resultado final da grade horária.

7. CONSIDERAÇÕES FINAIS

O problema de distribuição de grade horária é complexo por possuir diversas variáveis envolvidas, o método utilizado para a solução deste problema foi o Algoritmo Genético.

Embora trabalhosa a estruturação do AG, principalmente a estruturação do cromossomo vale a pena a análise, como o cromossomo foi bem estruturado ele foi capaz de já satisfazer algumas restrições impostas neste trabalho, o que facilitou muito no restante do desenvolvimento. Mostrando o quanto é importante a estruturação do cromossomo.

A escolha do operador de cruzamento por ordem, foi para que assim não existissem horários repetidos, o que causaria uma grade horária inválida. A mutação foi realizada apenas permutando horários dentro dos horários relacionados a turma, também para evitar resultados ineficazes.

A grade horária foi gerada corretamente e obteve diversos resultados satisfatórios, ou seja, sem infringir nenhuma restrição. Os resultados da simulação do AG se deram após 1000 gerações, durando aproximadamente 14 minutos a sua execução. A execução do AG se encerrou devido ao número de iterações estipuladas, sendo esse o critério de parada. Conclui-se então que a utilização do AG foi eficiente para a solução do problema.

Para a codificação do AG foi utilizada a linguagem Java, que é orientada a objetos e possui diversos recursos prontos. O banco de dados em MySQL possui uma boa comunicação com a linguagem Java e para o relacionamento entre banco de dados e sistema foi utilizado o Hibernate, facilitando a interação entre ambos.

Para trabalhos futuros podem ser implementados todas as restrições levantadas; também pode ser implementado outros operadores de recombinação e mutação para realizar comparações entre eles e escolher o melhor para o problema; existe a possibilidade também de fazer uma implementação em nuvem do banco de dados para que diversas instituições pudessem utilizar o sistema; para finalizar seria interessante fazer uma implementação gráfica para que um usuário final consiga usufruir do sistema.

REFERENCIAS

ANDRADE, Pedro Rochavetz de Lara. **Otimização na geração de Grade Horária Escolar através de um modelo matemático e das meta-heurísticas busca local e Iterated Local Search**. 2014. 222 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Universidade Federal do Paraná, Curitiba, 2014. Disponível em: <[http://acervodigital.ufpr.br/bitstream/handle/1884/35170/R - D - PEDRO ROCHAVETZ DE LARA ANDRADE.pdf?sequence=1](http://acervodigital.ufpr.br/bitstream/handle/1884/35170/R_-_D_-_PEDRO_ROCHAVETZ_DE_LARA_ANDRADE.pdf?sequence=1)>. Acesso em: 05 jul. 2016.

AUSTREGÉSILO, Márcio S. S. **Algoritmos Genéticos otimizando a distribuição de salas de aula na UFRPE**. 2014. 50 p. Monografia (Bacharel em Sistemas de Informação) – Universidade Federal Rural de Pernambuco, Recife, 2014. Disponível em: <<http://www.bsi.ufrpe.br/sites/www.bsi.ufrpe.br/files/Algoritmos%20geneticos%20otimizando%20a%20distribuicao%20de%20salas%20de%20aula%20na%20UFRPE.pdf>>. Acessado em: 28 maio 2016.

BEPPLER, Anderson. **Algoritmo Genético para geração de escala horária médica**. 2009. 91 p. Monografia (Bacharel em Sistemas de Informação, Universidade do Planalto Catarinense, Lages, 2009. Disponível em: <https://revista.uniplac.net/ojs/index.php/tc_si/article/viewFile/846/556>. Acesso em: 16 set. 2016.

BORGES, Suzana Kelly. **Resolução de Timetabling utilizando algoritmos genéticos e evolução cooperativa**. 2003. 104 f. Dissertação (Mestrado) - Curso de Informática, Universidade Federal do Paraná, Curitiba, 2003. Disponível em: <[http://acervodigital.ufpr.br/bitstream/handle/1884/25068/D - BORGES, SUZAN KELLY.pdf?sequence=1](http://acervodigital.ufpr.br/bitstream/handle/1884/25068/D_-_BORGES,_SUZAN_KELLY.pdf?sequence=1)>. Acesso em: 22 ago. 2016.

BRAZ JÚNIOR, Osmar O. L. **Otimização de horários em instituições de ensino superior através de algoritmos genéticos**. 2000. 144 p. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis, 2000.

GUEDES, Gilleanes T. A.. **UML 2: Uma Abordagem Prática**. São Paulo: Novatec Editora Ltda, 2009. 485 p.

NAKASHIMA, Elcio. **Algoritmos genéticos – um resumo e um exemplo**. 2009. Disponível em: <<https://tisimples.wordpress.com/2009/08/13/algoritmos-geneticos-um-resumo-e-um-exemplo/>>. Acesso em: 16 set. 2016.

HAMAWAKI, Cristiane D. L. **Geração Automática de Grade Horária Usando Algoritmos Genéticos: O Caso da Faculdade de Engenharia Elétrica da UFU**. 2005. 104 p. Dissertação (Mestrado em Ciências) – Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia, 2005.

HOLLAND, J. H. **Adaptation in Natural and Artificial System**. University of Michigan Press, 1975.

LIDEN, Ricardo. **Algoritmos Genéticos**. 3. ed. Rio de Janeiro: Ciência Moderna, 2012. 496 p.

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

SILVA, E. E da. **Otimização de estruturas de concreto armado utilizando algoritmos genéticos**. São Paulo, 2001. Dissertação (Mestrado em Engenharia de Estruturas). Universidade de São Paulo.

APÊNDICE

APÊNDICE A - SQL referente ao banco de dados

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0;
```

```
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
```

```
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
```

```
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET
latin1 COLLATE latin1_swedish_ci ;
```

```
CREATE SCHEMA IF NOT EXISTS `db_grade_horario` DEFAULT
CHARACTER SET latin1 ;
```

```
USE `mydb` ;
```

```
USE `db_grade_horario` ;
```

```
-----
```

```
-- Table `db_grade_horario`.`curso`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `db_grade_horario`.`curso` (
```

```
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
```

```
  `nome` VARCHAR(255) NOT NULL ,
```

```
  `duracao` INT(11) NOT NULL ,
```

```
  PRIMARY KEY (`id`))
```

```
ENGINE = InnoDB
```


AUTO_INCREMENT = 8

DEFAULT CHARACTER SET = latin1;

-- Table `db_grade_horario`.`semestre`

CREATE TABLE IF NOT EXISTS `db_grade_horario`.`semestre` (

`id` INT(11) NOT NULL AUTO_INCREMENT ,

`numero` INT(11) NOT NULL ,

`idCurso` INT(11) NOT NULL ,

PRIMARY KEY (`id`) ,

INDEX `fk_Semestre_Curso1_idx` (`idCurso` ASC) ,

CONSTRAINT `fk_Semestre_Curso1`

FOREIGN KEY (`idCurso`)

REFERENCES `db_grade_horario`.`curso` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

AUTO_INCREMENT = 43

DEFAULT CHARACTER SET = latin1;

```
-----  
-- Table `db_grade_horario`.`disciplina`  
-----
```

```
CREATE TABLE IF NOT EXISTS `db_grade_horario`.`disciplina` (  
  `id` INT(11) NOT NULL AUTO_INCREMENT ,  
  `nome` VARCHAR(45) NOT NULL ,  
  `creditos` INT(11) NOT NULL DEFAULT '2' ,  
  `idSemestre` INT(11) NOT NULL ,  
  PRIMARY KEY (`id`) ,  
  INDEX `fk_Disciplina_Semestre1_idx` (`idSemestre` ASC) ,  
  CONSTRAINT `fk_Disciplina_Semestre1`  
    FOREIGN KEY (`idSemestre` )  
    REFERENCES `db_grade_horario`.`semestre` (`id` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
  
AUTO_INCREMENT = 526  
  
DEFAULT CHARACTER SET = latin1;
```

```
-----  
-- Table `db_grade_horario`.`professor`  
-----
```

```
CREATE TABLE IF NOT EXISTS `db_grade_horario`.`professor` (  
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
```

```

`nome` VARCHAR(255) NOT NULL ,

PRIMARY KEY (`id` )

ENGINE = InnoDB

AUTO_INCREMENT = 125

DEFAULT CHARACTER SET = latin1;

-----

-- Table `db_grade_horario`.`atribuicao`

-----

CREATE TABLE IF NOT EXISTS `db_grade_horario`.`atribuicao` (

  `id` INT(11) NOT NULL AUTO_INCREMENT ,

  `idProfessor` INT(11) NOT NULL ,

  `idDisciplina` INT(11) NOT NULL ,

  PRIMARY KEY (`id` ) ,

  INDEX `fk_Atribuicao_Professor1_idx` (`idProfessor` ASC) ,

  INDEX `fk_Atribuicao_Disciplina1_idx` (`idDisciplina` ASC) ,

  CONSTRAINT `fk_Atribuicao_Disciplina1`

    FOREIGN KEY (`idDisciplina` )

    REFERENCES `db_grade_horario`.`disciplina` (`id` )

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

  CONSTRAINT `fk_Atribuicao_Professor1`

    FOREIGN KEY (`idProfessor` )

    REFERENCES `db_grade_horario`.`professor` (`id` )

```

```
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 402
DEFAULT CHARACTER SET = latin1;
-----
-- Table `db_grade_horario`.`horario`
-----
CREATE TABLE IF NOT EXISTS `db_grade_horario`.`horario` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `dia` INT(11) NOT NULL ,
  `inicio` VARCHAR(5) NOT NULL ,
  `fim` VARCHAR(5) NOT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB
AUTO_INCREMENT = 45
DEFAULT CHARACTER SET = latin1;
-----
-- Table `db_grade_horario`.`periodo`
-----
CREATE TABLE IF NOT EXISTS `db_grade_horario`.`periodo` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
```

```
`turno` VARCHAR(45) NOT NULL ,
```

```
PRIMARY KEY (`id` )
```

```
ENGINE = InnoDB
```

```
AUTO_INCREMENT = 4
```

```
DEFAULT CHARACTER SET = latin1;
```

```
-----
```

```
-- Table `db_grade_horario`.`cursoperiodo`
```

```
-----
```

```
CREATE TABLE IF NOT EXISTS `db_grade_horario`.`cursoperiodo` (
```

```
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
```

```
  `idPeriodo` INT(11) NOT NULL ,
```

```
  `idCurso` INT(11) NOT NULL ,
```

```
  PRIMARY KEY (`id` ) ,
```

```
  INDEX `fk_CursoPeriodo_Periodo1_idx` (`idPeriodo` ASC) ,
```

```
  INDEX `fk_CursoPeriodo_Curso1_idx` (`idCurso` ASC) ,
```

```
  CONSTRAINT `fk_CursoPeriodo_Curso1`
```

```
    FOREIGN KEY (`idCurso` )
```

```
    REFERENCES `db_grade_horario`.`curso` (`id` )
```

```
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
```

```
  CONSTRAINT `fk_CursoPeriodo_Periodo1`
```

```
    FOREIGN KEY (`idPeriodo` )
```

```
    REFERENCES `db_grade_horario`.`periodo` (`id` )
```

```
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 14
DEFAULT CHARACTER SET = latin1;

-----
-- Table `db_grade_horario`.`turma`
-----

CREATE TABLE IF NOT EXISTS `db_grade_horario`.`turma` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `inicio` DATE NOT NULL ,
  `idCursoPeriodo` INT(11) NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_Turma_CursoPeriodo1_idx` (`idCursoPeriodo` ASC) ,
  CONSTRAINT `fk_Turma_CursoPeriodo1`
    FOREIGN KEY (`idCursoPeriodo` )
    REFERENCES `db_grade_horario`.`cursoperiodo` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 67
DEFAULT CHARACTER SET = latin1;
```

```
-----  
-- Table `db_grade_horario`.`aula`  
-----
```

```
CREATE TABLE IF NOT EXISTS `db_grade_horario`.`aula` (  
  `id` INT(11) NOT NULL AUTO_INCREMENT ,  
  `idTurma` INT(11) NOT NULL ,  
  `idHorario` INT(11) NOT NULL ,  
  `idAtribuicao` INT(11) NOT NULL ,  
  PRIMARY KEY (`id`) ,  
  INDEX `fk_Aula_Horario1_idx` (`idHorario` ASC) ,  
  INDEX `fk_Aula_Atribuicao1_idx` (`idAtribuicao` ASC) ,  
  INDEX `fk_Aula_Turma1` (`idTurma` ASC) ,  
  CONSTRAINT `fk_Aula_Atribuicao1`  
    FOREIGN KEY (`idAtribuicao` )  
    REFERENCES `db_grade_horario`.`atribuicao` (`id` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Aula_Horario1`  
    FOREIGN KEY (`idHorario` )  
    REFERENCES `db_grade_horario`.`horario` (`id` )  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Aula_Turma1`  
    FOREIGN KEY (`idTurma` )
```

```

REFERENCES `db_grade_horario`.`turma` (`id` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = latin1;

-----

-- Table `db_grade_horario`.`cursoperiodohorario`

-----

CREATE TABLE IF NOT EXISTS `db_grade_horario`.`cursoperiodohorario` (

  `idCursoPeriodoHorario` INT(11) NOT NULL AUTO_INCREMENT ,

  `idCursoPeriodo` INT(11) NOT NULL ,

  `idHorario` INT(11) NOT NULL ,

  PRIMARY KEY (`idCursoPeriodoHorario`) ,

  INDEX `fk_idCursoPeriodo_idx` (`idCursoPeriodo` ASC) ,

  INDEX `fk_horario_idx` (`idHorario` ASC) ,

  CONSTRAINT `fk_horario`

    FOREIGN KEY (`idHorario` )

      REFERENCES `db_grade_horario`.`horario` (`id` )

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

  CONSTRAINT `fk_idCursoPeriodo`

    FOREIGN KEY (`idCursoPeriodo` )

      REFERENCES `db_grade_horario`.`cursoperiodo` (`id` )

```



```

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

AUTO_INCREMENT = 195

DEFAULT CHARACTER SET = latin1;

-----

-- Table `db_grade_horario`.`disponibilidade`
-----

CREATE TABLE IF NOT EXISTS `db_grade_horario`.`disponibilidade` (
  `id` INT(11) NOT NULL AUTO_INCREMENT ,
  `disponivel` TINYINT(1) NOT NULL DEFAULT '0' ,
  `idHorario` INT(11) NOT NULL ,
  `idProfessor` INT(11) NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_Disponibilidade_Horario1_idx` (`idHorario` ASC) ,
  INDEX `fk_Disponibilidade_Professor1_idx` (`idProfessor` ASC) ,
  CONSTRAINT `fk_Disponibilidade_Horario1`
    FOREIGN KEY (`idHorario` )
    REFERENCES `db_grade_horario`.`horario` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Disponibilidade_Professor1`
    FOREIGN KEY (`idProfessor` )

```

```
REFERENCES `db_grade_horario`.`professor` (`id` )  
  
ON DELETE NO ACTION  
  
ON UPDATE NO ACTION)  
  
ENGINE = InnoDB  
  
AUTO_INCREMENT = 16  
  
DEFAULT CHARACTER SET = latin1;  
  
  
SET SQL_MODE=@OLD_SQL_MODE;  
  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```