

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ  
DEPARTAMENTO DE COMPUTAÇÃO  
CIÊNCIA DA COMPUTAÇÃO**

**COMPUTAÇÃO EVOLUCIONÁRIA PARA ELABORAÇÃO DE  
QUADROS DE HORÁRIOS COMPLEXOS**

**ALAN RABELO MARTINS**

**MARACANAÚ**

**2019**

**ALAN RABELO MARTINS**

**COMPUTAÇÃO EVOLUCIONÁRIA PARA ELABORAÇÃO DE QUADROS DE  
HORÁRIOS COMPLEXOS**

Trabalho de Conclusão de Curso apresentado  
como requisito parcial para obtenção do grau de  
Bacharelado em Ciência da Computação da Instituto  
Federal de Educação, Ciência e Tecnologia do Ceará  
- IFCE, Departamento de Computação.

Orientador: Prof. Dr. Ajalmar R. R. Neto

**ALAN RABELO MARTINS**

**COMPUTAÇÃO EVOLUCIONÁRIA PARA ELABORAÇÃO DE QUADROS DE  
HORÁRIOS COMPLEXOS**

Trabalho de Conclusão de Curso apresentado  
como requisito parcial para obtenção do grau de  
Bacharelado em Ciência da Computação do Instituto  
Federal de Educação, Ciência e Tecnologia do Ceará  
- IFCE, Departamento de Computação.

Data de Aprovação: 22/08/2019

**Banca Examinadora**

---

Prof. Dr. Ajalmar R. R. Neto - IFCE - Campus Maracanaú  
Orientador

---

Prof. - IFCE- Campus Maracanaú  
Membro da Banca

---

Prof. Me. - IFCE- Campus Maracanaú  
Membro da Banca

Dedico este trabalho à minha família.

## **AGRADECIMENTOS**

Primeiramente a Deus, pois sem sua permissão nada seria possível. A meu pais Audir e Ivoneide, irmãos Adriana, Adriano e Júnior, por me motivarem a estudar desde cedo e sempre perseguir o caminho da ciência. À minha namorada Ruana por estar sempre comigo enfrentando todos os desafios da minha graduação. Ao Professor Ajalmar Rocha pelas aulas, orientações e incentivos, fundamentais para execução deste trabalho. Aos meus colegas de curso que contribuíram na realização deste trabalho. A todos os meus amigos que me incentivaram e me apoiaram. A todos os professores, colegas e funcionários do IFCE que tiveram papel importante na minha formação profissional e pessoal.

Não sei o que possa parecer aos olhos do mundo, mas aos meus pareço apenas ter sido como um menino brincando à beira-mar, divertindo-me com o fato de encontrar de vez em quando um seixo mais liso ou uma concha mais bonita que o normal, enquanto o grande oceano da verdade permanece completamente por descobrir à minha frente.

Isaac Newton

## RESUMO

Problemas de quadro de horários estão presentes em diversas empresas e instituições e consiste, de forma geral, na alocação de um conjunto de recursos em *slots* de tempo. Isso claramente se reflete em instituições educacionais. Atualmente a organização de quadros de horários como esse, principalmente para instituições com um número maior de professores, salas, cursos e, conseqüentemente, disciplinas necessita de uma demanda muito grande de tempo e trabalho manual. Muito se tem estudado e vários métodos foram desenvolvidos para tentar automatizar a geração de quadros de horários, que é classificada como NP-Difícil. Este trabalho se propõe a solucionar o problema utilizando Algoritmos Genéticos para encontrar soluções ótimas ou quase ótimas para problemas como este além de criar uma plataforma Web e uma API REST que auxilia o usuário no fornecimento de dados e parâmetros necessários para execução do algoritmo. Foi avaliado, neste trabalho, a aplicação do sistema no Instituto Federal do Ceará - Campus Maracanaú.

**Palavras-chave:** Problema do Quadro de Horário de Aulas; Algoritmos Genéticos; Inteligência Artificial.

## **ABSTRACT**

Timetable problems are present in various companies and institutions and consist in allocating a set of resources in time slots. This seems to show up in educational institutions. Currently, the organization of such time frames, especially for institutions with a larger number of professors, courses and, consequently, long-term subjects and manual work. A lot has been studied and several methods were developed to try to automate a generation of schedules, which is classified as NP-Difficult. This work proposes the problems of problem using Algorithms Genetic through a web platform and a REST API using the user of data and methods required to the execution of algorithm. An application of this system was evaluated in the Federal Institute of Ceará - Campus Maracanaú.

**Keywords:** Schedule Timetabling Problem; Genetics Algorithms; Artificial Intelligence.



## LISTA DE FIGURAS

Figura 1 – Diagrama de Execução de um algoritmo genético . . . . .	22
Figura 2 – Cromossomo que representa o problema . . . . .	26
Figura 3 – Operador de crossover aplicado nos sub-níveis de semestres . . . . .	27
Figura 4 – Operador de mutação swap mutation restrito a operar em um semestre . . . . .	27
Figura 5 – Lista de professores . . . . .	30
Figura 6 – Tela de adicionar um novo professor . . . . .	30
Figura 7 – Adicionando indisponibilidade de horários do professor . . . . .	30
Figura 8 – Módulo Responsável por adicionar eixos, cursos e salas . . . . .	31
Figura 9 – Lista de Eixos . . . . .	31
Figura 10 – Lista de Cursos . . . . .	31
Figura 11 – Lista de Disciplinas . . . . .	32
Figura 12 – Geração de Quadro de Horários . . . . .	33
Figura 13 – Lista exibindo a quantidade de choques . . . . .	33
Figura 14 – Exemplo de URI . . . . .	35
Figura 15 – Tela inicial de um Web app na azure . . . . .	38
Figura 16 – Azure Deployment Center . . . . .	39
Figura 17 – Método responsável pela obtenção do JSON a partir da requisição recebida . . . . .	44
Figura 18 – JSON retornado pelo sistema . . . . .	45

## **LISTA DE ABREVIATURAS E SIGLAS**

JGAP	Java Genetic Algorithms Package
AG	Algoritmos Genéticos
PQH	Problema de Quadro de Horário
API	Application Programming Interface
REST	Representational State Transfer
HTTP	HyperText Transfer Protocol
JPA	Java Persistence API
JSF	JavaServer Faces

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>14</b>
1.1	Justificativa . . . . .	15
1.2	Objetivos . . . . .	16
1.2.1	Organização . . . . .	16
<b>2</b>	<b>ALGORITMOS GENÉTICOS E APLICAÇÃO AO PROBLEMA . . .</b>	<b>17</b>
2.1	Computação Evolucionária . . . . .	17
2.2	Elementos de Algoritmos Genéticos . . . . .	18
2.2.1	População . . . . .	18
2.2.2	Cromossomos e Funções de Fitness . . . . .	19
2.2.3	Operadores Genéticos . . . . .	19
2.2.3.1	Seleção . . . . .	20
2.2.3.2	Crossover . . . . .	20
2.2.3.3	Elitismo . . . . .	20
2.2.3.4	Mutação . . . . .	21
2.2.4	Função de Fitness . . . . .	21
2.2.5	Execução do Algoritmo . . . . .	21
2.3	Algoritmos Genéticos aplicados ao problema de quadro de horário . . . . .	24
2.3.1	Restrições . . . . .	24
2.3.1.1	Restrições hard . . . . .	24
2.3.2	Restrições soft . . . . .	24
2.3.3	Representação do Problema . . . . .	25
2.3.4	Operadores genéticos aplicados ao problema . . . . .	26
2.3.4.1	Seleção . . . . .	26
2.3.4.2	Crossover . . . . .	26
2.3.4.3	Mutação . . . . .	27
2.3.4.4	Função de aptidão (fitness) . . . . .	27
<b>3</b>	<b>SISTEMA WEB E API REST . . . . .</b>	<b>29</b>
3.1	Sistema WEB . . . . .	29
3.1.1	Professores . . . . .	29
3.1.2	Eixos, cursos e disciplinas . . . . .	31
3.1.3	Ofertas . . . . .	32
3.2	Geração do Quadro de Horários . . . . .	32

3.2.1	Tela de Geração e Exibição . . . . .	32
3.3	API Rest . . . . .	34
3.3.1	Modelo cliente/servidor . . . . .	34
3.3.2	Stateless . . . . .	34
3.3.3	Interface Uniforme . . . . .	35
3.3.4	Cache . . . . .	35
3.4	Restful . . . . .	35
3.4.1	Endereçamento . . . . .	35
3.4.2	Métodos Http . . . . .	36
3.4.3	Stateless . . . . .	36
3.5	Aplicação . . . . .	36
3.5.1	Azure . . . . .	36
3.5.2	Entrada de dados . . . . .	39
3.5.3	Execução . . . . .	44
<b>4</b>	<b>RESULTADOS DA APLICAÇÃO . . . . .</b>	<b>46</b>
<b>5</b>	<b>CONCLUSÃO E TRABALHOS FUTUROS . . . . .</b>	<b>48</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>49</b>

## 1 INTRODUÇÃO

O Problema de Alocação de Recursos, em nosso caso específico Quadro de Horários, propõe alocar um conjunto de professores em um conjunto de horários respeitando restrições impostas pelo usuário. As restrições podem ser obrigatórias ou opcionais. Restrições Obrigatórias levam em conta a inviabilidade da solução final. Um mesmo professor não pode, de forma alguma, ser escalado em duas disciplinas no mesmo horário. Da mesma forma, disciplinas do mesmo semestre não devem estar no mesmo horário nem a mesma sala pode ser alocada para duas disciplinas no mesmo período. Restrições optativas no entanto não representam uma infactibilidade, mas uma maneira de classificar resultados factíveis como melhores ou piores. Podemos citar, por exemplo, o caso de disciplinas geminadas, em que uma mesma disciplina foi alocada de uma forma em que sua carga horária semanal está contida em um só dia. Como restrição optativa podemos tomar como exemplo também o caso de um professor estar alocado em um horário pré-estabelecido por ele como indisponível.

Uma solução para este tipo de problema consiste na geração de quadros de horários que minimize a violação das restrições, e se dá através do uso de ferramentas de otimização (RODRIGUEZ et al., 2014). Este problema é um problema chamado NP-difícil. Ele possui tem dificuldade de ter uma solução encontrada dentro de um prazo razoável devido ao elevado grau de dificuldade para solucioná-lo.

Na Inteligência Artificial podemos encontrar várias ferramentas para solução de problemas de otimização. A utilização de Algoritmos genéticos são frequentemente propostas para problemas desse tipo. Algoritmos Genéticos, juntamente como Programação Genética formam uma sub-área da Inteligência Artificial conhecida como Computação Evolucionária. Esta sub-área, composta por algoritmos chamados de bio-inspirados utiliza uma abordagem parecida com as leis de evolução biológica de Darwin e reprodução de espécies de Mendel. Nesses tipos de algoritmos, assim como na vida real, várias soluções são criadas e competem pela "sobrevivência" ao serem avaliados através de meta-heurísticas, enquanto sofrem mutações aleatórias e se reproduzem, gerando novas soluções resultantes da combinação de duas soluções distintas.

Vale ressaltar que esse problema pode ser aplicado em qualquer instância da educação em que a rotação de professores seja utilizada, ou seja, cada professor ministra um conjunto de disciplinas. Tendo assim como sua gama de utilizadores, Universidades públicas e particulares, escolas de ensino fundamental e médio. Sua aplicação no Instituto Federal do Ceará será execu-

tada a partir da entrada de informações sobre cursos, disciplinas, professores e indisponibilidades além de parâmetros de configuração. Todas as informações serão obtidas a partir da interface Web implementada.

## 1.1 JUSTIFICATIVA

Vários métodos foram aplicados para tentar resolver o problema em questão. No início das pesquisas nessa área, métodos utilizando teoria dos grafos representavam o estado da arte utilizando técnicas como coloração de grafos eram utilizadas para resolver os problemas. Foi utilizado também seleção sequencial onde as tuplas eram escaladas uma a uma. Outras técnicas incluem programação linear e métodos meta-heurísticos como a têmpera simulada. Algoritmos genéticos foram amplamente utilizados em diferentes aspectos de problemas de quadros de horários. Foram investigados: Computação paralela, algoritmos gulosos, associação com grafos. Schaerf aplicou um algoritmo chamado *Tabu search* para resolver o problema em uma escola italiana. Foi utilizada uma matriz bi-dimensional inicializada aleatoriamente e são utilizados movimentos para tentar encontrar um quadro factível (que não viola nenhuma restrição).

(SMITH; DUKE, 2003) utilizou a têmpera simulada, técnica de inteligência artificial, para resolver o mesmo problema em uma escola australiana. Nesse caso uma solução candidata é gerada e as tuplas movem-se livremente enquanto o custo diminui e de acordo com que o algoritmo piora as tuplas ficam 'congeladas'. (Beligiannis et al, 2013). resolveram o problema para uma escola grega utilizando uma população de partículas (soluções candidatas), que são evoluídas ou melhoradas de acordo com o número de gerações. Partículas que não acompanham o ritmo de evolução são removidas.

A ausência de uma ferramenta computacional para auxiliar a elaboração de quadros de horários faz com que esta tarefa consuma vários dias, as vezes, semanas de esforço de um ou mais funcionários, incluindo o coordenador do curso, que ficam encarregados desta função, (MOUSA; EL-SISI, 2013). Automatizar a geração dos horários vem sendo uma necessidade, pois a solução manual deste problema é complexa e geralmente requer vários dias de trabalho, (ABOUELHAMAYED et al., 2016). A solução manual deste problema baseia-se em um processo de tentativa e erro, muitas vezes produzindo quadros de horários de baixa qualidade (ABOUELHAMAYED et al., 2016).

Já existem soluções para desenvolvimento dos quadros de horários, no entanto, pelo fato de buscarem uma generalização, não são adequadas aos ambientes específicos. Este trabalho propõe um sistema Web automatizado com ampliação de restrições e direcionado para instituições

educacionais a fim de otimizar a geração das tabelas de horários no Instituto Federal do Ceará - Campus Maracanaú. A ideia visa atender a demanda de geração de esquemas de horários em todos os cursos, reduzindo assim gastos financeiros e de tempo na elaboração destes quadros.

## 1.2 OBJETIVOS

Este trabalho tem por objetivo o desenvolvimento de um sistema WEB e uma API REST inteligente capazes de gerar automaticamente o quadro de horários de aulas que atenda todas as restrições que serão identificadas no Instituto Federal do Ceará (IFCE), a partir de parâmetros fornecidos pelo usuário. Esse sistema poderá ser aplicado a outras instituições da comunidade, como escolas públicas municipais ou estaduais além de outras universidades. Além disso visa diminuir o tempo necessário para elaboração destes quadros de horários.

Além deste objetivo geral tem-se os seguintes objetivos específicos:

1. Compreender o problema do quadro de horários
2. Compreender e aplicar conceitos de algoritmos genéticos para a solução do problema
3. Propor um sistema WEB e uma API Rest para geração automática de quadros de horários

### 1.2.1 Organização

O presente documento está organizado da seguinte forma: no Capítulo 2 introduz conceitos de algoritmos genéticos e os aspectos mais importantes referentes aos mesmos. No Capítulo 3 é apresentado o problema de quadros de horários, suas variações e características. No capítulo 4 apresento a aplicação de algoritmos genéticos ao problema em questão. No capítulo 5 é exibido os resultados do desenvolvimento da aplicação WEB e da API. No capítulo 6 apresentamos a conclusão e trabalhos futuros.

Foi utilizada a linguagem Python, no ambiente de desenvolvimento Pycharm. O ambiente completo de API e Web service se baseia em Flask, um framework para aplicações web em Python.

## 2 ALGORITMOS GENÉTICOS E APLICAÇÃO AO PROBLEMA

### 2.1 COMPUTAÇÃO EVOLUCIONÁRIA

De acordo com Mitchell (1996), nos anos 50 e 60, vários cientistas da computação estudaram independentemente sistemas evolucionários com a ideia de que a evolução poderia ser utilizada como uma ferramenta de otimização para problemas de engenharia. A ideia de todos esses sistemas era evoluir uma população de soluções candidatas para um problema dado utilizando operadores inspirados por variação genética e seleção natural.

Algoritmos genéticos (AGs) foram inventados por John Holland na década de 1960 e foram desenvolvidos por Holland e seus alunos e colegas da Universidade de Michigan nos anos 1960 e 1970. Em contraste com as estratégias de evolução e programação evolutiva, o objetivo original de Holland não era projetar algoritmos para resolver problemas específicos, mas sim estudar formalmente o fenômeno da adaptação como ocorre na natureza e desenvolver maneiras pelas quais os mecanismos de adaptação natural pudessem ser importados em sistemas de computador.

AGs utilizam uma analogia com termos da genética. De acordo com Mitchell (1996):

- Genes são as características que representam os indivíduos;
- Cromossomos são compostos por um aglomerado de genes que juntos representam uma parte da solução ou a própria solução do problema de busca. Um cromossomo é formado por  $N$  genes que possuem valor e posição;
- Alelo é o valor atribuído ao gene;
- Locus é a posição do gene no cromossomo;
- Genótipo é a estrutura da solução, é composto por um ou mais cromossomos;
- Fenótipo é a decodificação da estrutura do genótipo;
- Indivíduo representa um estado do espaço de busca. Quando o indivíduo tem apenas um cromossomo, cromossomo e indivíduo são tratados indistintamente;
- Ambiente é o problema, ou seja, o espaço de busca em que se deseja encontrar um estado que maximize uma determinada função objetivo (ou minimize uma determinada função custo);



- Aptidão Fitness é o grau de aptidão de um indivíduo ao ambiente, ou seja, o valor obtido da função de aptidão.

Mitchell (1996) explica que a Adaptação em Sistemas Naturais e Artificiais, de 1975, de Holland, apresentou o algoritmo genético como uma abstração da evolução biológica e forneceu um arcabouço teórico para adaptação sob o GA. GA de Holland é um método para passar de uma população de "cromossomos"(por exemplo, cadeias de uns e zeros, ou "bits") para uma nova população usando uma espécie de "seleção natural"junto com os operadores de crossover inspirados na genética. , mutação e inversão. Cada cromossomo consiste em "genes"(por exemplo, bits), cada gene sendo uma instância de um "alelo"particular (por exemplo, 0 ou 1). O operador de seleção escolhe os cromossomos da população que será permitida a reprodução e, em média, os cromossomos mais aptos produzem mais descendentes do que os cromossomos mais aptos. Crossover troca subpartes de dois cromossomos, imitando aproximadamente a recombinação biológica entre dois organismos de cromossomo único ("haploid"); mutação muda aleatoriamente os valores alélicos de alguns locais no cromossomo; e a inversão inverte a ordem de uma seção contígua do cromossomo, rearranjando, assim, a ordem na qual os genes são organizados. (Aqui, como na maior parte da literatura do GA, "cruzamento"e "recombinação"significarão a mesma coisa.)

## 2.2 ELEMENTOS DE ALGORITMOS GENÉTICOS

De acordo com Mitchell (1996) não existe uma definição rigorosa de "algoritmo genético"aceita por todos na comunidade de computação evolucionária que diferencia GAs de outros métodos de computação evolucionários. No entanto, pode-se dizer que a maioria dos métodos chamados "AGs"tem pelo menos os seguintes elementos em comum: populações de cromossomos, seleção de acordo com a aptidão, cruzamento para produzir novos descendentes e mutação aleatória de novos descendentes.Inversão - o quarto elemento de Holland GAs - raramente são usados nas implementações de hoje, e suas vantagens, se houver, não estão bem estabelecidas. (A inversão será discutida em detalhes no capítulo 5.)

### 2.2.1 População

A população é um conjunto de indivíduos candidatos a participarem do processo de seleção, reprodução e mutação. Geralmente a população inicial é gerada aleatoriamente fornecendo

maior biodiversidade e abrangência do espaço de busca. Os operadores de inicialização mais comuns são (GOLDBERG, 1989):

- Inicialização heurística: os indivíduos são criados a partir de heurísticas. No entanto, há a possibilidade de gerar indivíduos semelhantes prejudicando a diversidade da população, para este caso, adotam-se métodos aleatórios em conjunto com heurísticas;
- Inicialização aleatória não uniforme: um conjunto de alelos tendem a ser escolhidos com maior frequência;
- Inicialização aleatória com *dope*: indivíduos mais aptos são inseridos na população gerada aleatoriamente. No entanto, há o risco destes indivíduos dominarem o processo de evolução acarretando o problema da convergência prematura (problema onde os filhos são muito semelhantes aos pais ou até mesmo estagnação das evoluções);
- Inicialização aleatória uniforme: o gene recebe um valor sorteado aleatoriamente de forma uniforme de um conjunto de alelos.

O tamanho da população é uma das mais importantes escolhas que deve ser levada em consideração ao utilizar AGs, pois uma população pequena pode gerar soluções ruins em muitas aplicações. Além disto, se a população for muito pequena, os AGs tendem a convergir rapidamente; se for muito grande, há um desperdício de recursos computacionais, pois o tempo de espera para alcançar indivíduos mais aptos pode ser muito longo (MICHALEWICZ, 1992).

### 2.2.2 Cromossomos e Funções de Fitness

Os cromossomos em uma população GA normalmente tomam a forma de sequências de bits. Cada locus no cromossomo tem dois alelos possíveis: 0 e 1. Cada cromossomo pode ser considerado como um ponto no espaço de busca de soluções candidatas. O GA processa populações de cromossomos, substituindo sucessivamente uma dessas populações por outra. O GA geralmente requer uma função de aptidão que atribui uma pontuação (aptidão) a cada cromossomo na população atual. A adequação de um cromossomo depende de quão bem esse cromossomo resolve o problema em questão.

### 2.2.3 Operadores Genéticos

A forma mais simples de algoritmo genético envolve três tipos de operadores que são classificados como: seleção, cruzamento e mutação. Estes operadores são utilizados pelo algoritmo

para, através das gerações, aumentar a qualidade de suas soluções.

#### 2.2.3.1 Seleção

Este operador é responsável por escolher quais indivíduos serão utilizados na geração de novos descendentes. Segundo (GOLDBERG, 1989) os métodos mais conhecidos para a tarefa de seleção são:

- **Roleta:** cada indivíduo recebe na roleta uma fatia correspondente ao seu valor de aptidão (fitness), ou seja, indivíduos mais aptos recebem uma porção maior da roleta tendo mais probabilidade de serem selecionados, e os menos aptos uma porção menor. A roleta é girada um determinado número de vezes, de tal maneira que a cada giro um indivíduo é selecionado. O genoma dos indivíduos sorteados podem participar da próxima geração;
- **Amostragem Universal Estocástica (Stochastic Universal Sampling - SUS):** semelhante à roleta, mas para selecionar  $k$  indivíduos utiliza  $k$  agulhas igualmente espaçadas, girando-as em conjunto uma só vez;
- **Torneio:** funciona similarmente a um torneio, dois indivíduos são sorteados e o mais apto vence, assim, todos que vencerem o torneio serão os selecionados pelo método.

#### 2.2.3.2 Crossover

Este operador, conhecido também como operador de cruzamento escolhe aleatoriamente um locus e troca as subsequências antes e depois desse locus entre dois cromossomos para criar dois descendentes. Por exemplo, as cadeias 10000100 e 11111111 poderiam ser cruzadas após o terceiro locus em cada um para produzir os dois descendentes 10011111 e 11100100.

O operador de cruzamento imita aproximadamente a recombinação biológica entre dois organismos cromossomo-simples (haplóides) e gera novos indivíduos que formarão a próxima geração, além de atuar com uma probabilidade dada pela taxa de cruzamento  $P_c$ . Assim, considerando as diversas combinações possíveis das características dos pais, um número bastante elevado de soluções podem ser geradas. Em AGs, as implementações de *crossover*, em geral, produzem dois novos indivíduos a partir de outros dois.

#### 2.2.3.3 Elitismo

O elitismo é uma operação utilizada para evitar que soluções ótimas ou quase ótimas sejam descartadas na fase de reprodução ou mutação. Seu funcionamento garante os melhores

indivíduos da geração atual na próxima.

#### 2.2.3.4 Mutação

Os operadores de mutação são responsáveis por manter a diversidade genética na população e, conseqüentemente, explorar novos espaços de busca. Esta etapa ocorre após a reprodução. Ele modifica aleatoriamente um ou mais genes de um cromossomo. A probabilidade de ocorrência em um gene é dada pela taxa de mutação. O operador percorre cada gene do cromossomo gerando um evento de probabilidade  $P_m$ ; Usualmente as taxas de mutação recebem valores muito pequenos variando de 0 a 1, e os mais utilizados são 0,001 e 0,01. A taxa de mutação não pode ser muito alta, pois torna o algoritmo basicamente aleatório.

(GOLDBERG, 1989) nos mostra exemplos de algoritmos de mutação:

- **Mutação aleatória (flip mutation):** cada gene mutado recebe um valor sorteado pertencente ao alfabeto válido (entenda como alfabeto válido, o universo no qual os valores dos genes estão contidos, inteiros, string, binários, etc);
- **Mutação por troca (swap mutation):** são sorteados  $n$  pares de genes e os pares de genes trocam os valores entre si;
- **Mutação creep:** um valor aleatório é somado ou subtraído do valor do gene.

#### 2.2.4 Função de Fitness

A função de *fitness* ou função de aptidão é responsável por retornar o grau de aptidão de um indivíduo no intervalo entre 0 e 1 (Sendo 1 mais apto e 0 menos apto), avaliando sua qualidade enquanto solução do problema. Nesta função são embutidas todas as restrições que o algoritmo deve satisfazer além de seu objetivo de qualidade.

#### 2.2.5 Execução do Algoritmo

1. Comece com uma população gerada aleatoriamente de cromossomos de  $n$ -bits (soluções candidatas a um problema).
2. Calcule a aptidão  $f(x)$  de cada cromossomo  $x$  na população.
3. Repita os seguintes passos até que  $n$  descendentes tenham sido criados:
  - a) Selecione um par de cromossomos pai da população atual, a probabilidade de seleção ser uma função crescente de aptidão. A seleção é feita "com substituição", o que

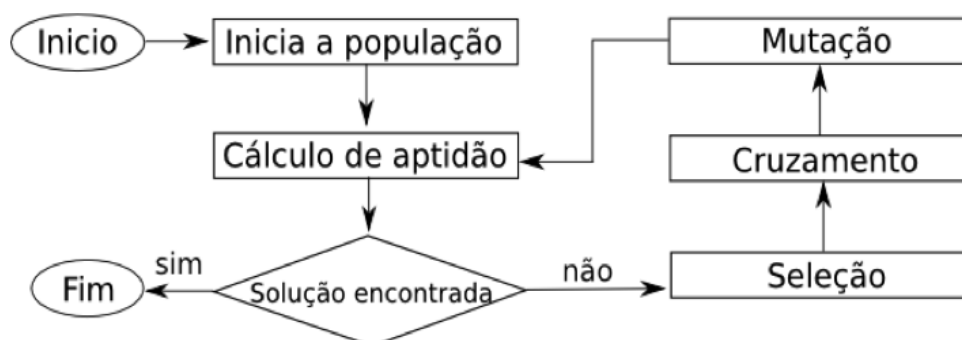
significa que o mesmo cromossomo pode ser selecionado mais de uma vez para se tornar pai.

- b) Com probabilidade  $pc$  (a "probabilidade de cruzamento" ou "taxa de cruzamento"), cruze o par em um ponto escolhido aleatoriamente (escolhido com probabilidade uniforme) para formar dois descendentes. Se não houver cruzamento, formar dois descendentes que sejam cópias exatas de seus respectivos pais. (Observe que aqui a taxa de crossover é definida como a probabilidade de que dois pais cruzarão em um único ponto. Há também versões "cross-point crossover" do GA em que a taxa de crossover para um par de pais é o número de pontos nos quais um cruzamento ocorre.)
- c) Mude os dois descendentes em cada locus com probabilidade  $pm$  (a probabilidade de mutação ou taxa de mutação) e coloque os cromossomos resultantes na nova população. Se  $n$  for ímpar, um novo membro da população pode ser descartado aleatoriamente.

4. Substitua a população atual pela nova população.

5. Vá para o passo 2.

Figura 1 – Diagrama de Execução de um algoritmo genético



Cada iteração desse processo é chamada de geração. Um GA é tipicamente iterado para qualquer lugar entre 50 e 500 ou mais gerações. Todo o conjunto de gerações é chamado de corrida. No final de uma corrida, muitas vezes há um ou mais cromossomos altamente adequados na população. Como a aleatoriedade desempenha um grande papel em cada execução, duas execuções com diferentes sementes de números aleatórios geralmente produzem diferentes comportamentos detalhados. Pesquisadores GA geralmente relatam estatísticas (como a melhor aptidão encontrada em uma corrida e a geração na qual o indivíduo com a melhor aptidão

física foi descoberto) calculou a média em muitas execuções diferentes da GA sobre o mesmo problema. (MITCHELL, 1996)

O procedimento simples que acabamos de descrever é a base para a maioria das aplicações dos AGs. Há vários detalhes a serem preenchidos, como o tamanho da população e as probabilidades de cruzamento e mutação, e o sucesso do algoritmo geralmente depende muito desses detalhes. Há também versões mais complicadas de AGs (por exemplo, GAs que trabalham em representações diferentes de strings ou GAs que possuem tipos diferentes de operadores de crossover e mutação). Muitos exemplos serão dados em capítulos posteriores.

## 2.3 ALGORITMOS GENÉTICOS APLICADOS AO PROBLEMA DE QUADRO DE HORÁRIO

Assim como no estudo de caso de Alves et al. (2015), visamos gerar o quadro de horários para o cursos do Departamento de Computação do IFCE - Campus Maracanaú, sendo composto de três cursos: Ciência da Computação, nos turno matutino e vespertino, Técnico em Informática no turno vespertino e Técnico em Redes no turno matutino. Alves et al. (2015) descreve que o curso de Ciência da Computação tem a duração de 4 anos divididos em 8 semestres. Cada semestre poderá ter até 5 dias de aulas semanais, representando a semana, e cada dia é composto por 4 períodos a serem ocupados com uma relação professor/disciplina. Os cursos técnicos também são de 5 dias de aulas semanais e 4 períodos diários, a duração de ambos os cursos é de 2 anos totalizando 4 semestres. A carga horária das disciplinas é dada em valor de crédito, onde o número de créditos indica a sua ocorrência na grade semanal, ou seja, uma disciplina X de 4 créditos preenche 4 tempos na grade semanal.

### 2.3.1 Restrições

#### 2.3.1.1 Restrições hard

Para este trabalho utilizaremos três restrições obrigatórias, ou seja, caso alguma dessas restrições seja violada, a solução passa a ser não factível. Elas são:

- Sobreposição de aulas de um mesmo professor: Com essa restrição um professor não pode ministrar duas disciplinas diferentes no mesmo horários.
- Sobreposição de disciplinas do mesmo semestre: Um mesmo semestre não pode ter choque de horários em suas disciplinas, caso contrário seria inviável aos alunos cursar todas as disciplinas do mesmo.
- Sobreposição de salas em um mesmo horário: Cada disciplina é atrelada a uma sala/laboratório. Dessa forma é inviável que duas disciplinas alocadas na mesma sala/laboratório sejam alocados no mesmo período de tempo pois duas aulas diferentes não podem ocorrer no mesmo horário e sala.

### 2.3.2 Restrições soft

Temos duas restrições optativas, que caso violadas não representam uma invalidação da solução mas caso cumpridas torna a solução ótima. Elas são:

- Evitar disciplinas geminadas: uma mesma disciplina não pode se concentrar inteiramente em um dia, para que assim, em uma disciplina de 4 créditos, 4 períodos consecutivos de um mesmo dia sejam ocupados.
- Indisponibilidade dos professores: No sistema desenvolvido, professores podem indicar suas indisponibilidades de horários, devido à realização de outras atividades.

Tabela	Segunda	Terça	Quarta	Quinta	Sexta
8:00 às 10:00 (AB Manhã)	0	1	2	3	4
10:00 às 12:00 (CD Manhã)	5	6	7	8	9
14:00 às 16:00 (AB Tarde)	10	11	12	13	14
16:00 às 18:00 (CD Tarde)	15	16	17	18	19

Tabela 1 – Representação dos slots de tempo de acordo com dia da semana de período de horário. Exemplo para manhã e tarde

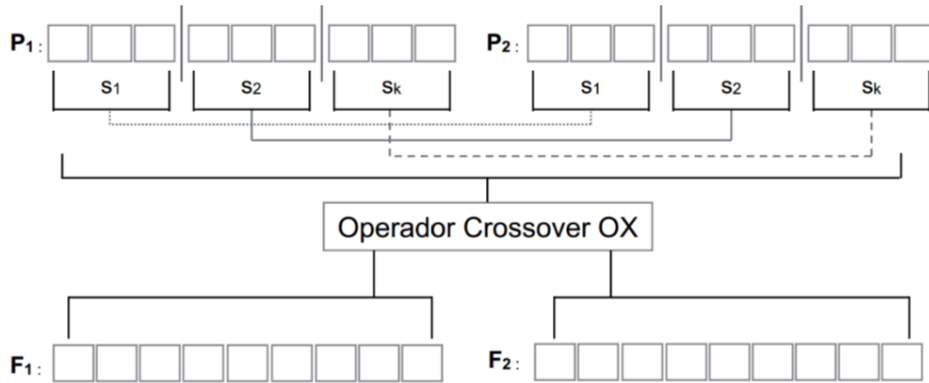
### 2.3.3 Representação do Problema

Para a representação do problema a ser tratado utilizaremos a mesma abordagem de Alves et al. (2015) que considera o cromossomo sendo formado por genes do tipo inteiro, cujo valores não se repetem, seu tamanho é variável o qual depende do número de semestres e períodos diários. Cada gene recebe um valor inteiro de identificação único (ID) que está relacionado a uma instância de uma relação professor/disciplina/sala. Alves et al. (2015) também conclui que um gene representa um slot na grade horária semanal, o qual equivale a 2 períodos, que podem estar vazios indicando que não há aula naqueles períodos, ou representar 1 ou 2 créditos de uma relação professor/disciplina/sala.





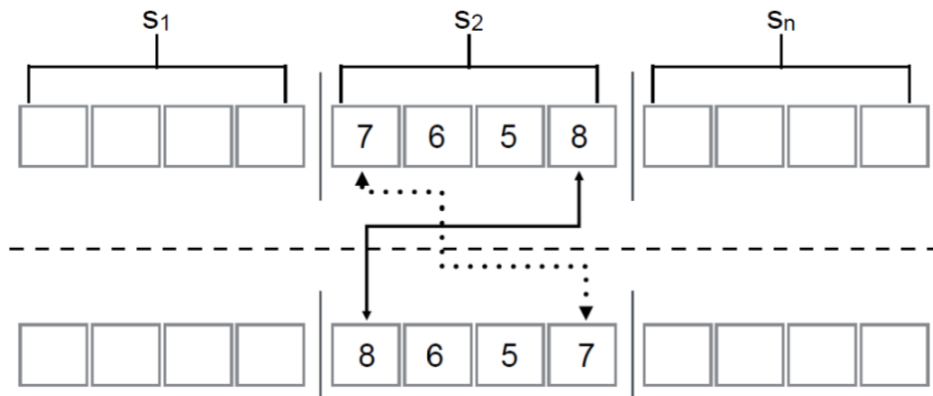
Figura 3 – Operador de crossover aplicado nos sub-níveis de semestres



#### 2.3.4.3 Mutação

Como operador de mutação foi escolhido o swap mutation, dessa forma, o operador efetua a troca de duas posições no cromossomo. Vale ressaltar que assim como no operador OX, temos que garantir que todas as trocas sejam efetuadas dentro do semestre, pois caso sejam trocados disciplinas de dois semestres diferentes, o cromossomo se tornaria inválido.

Figura 4 – Operador de mutação swap mutation restrito a operar em um semestre



#### 2.3.4.4 Função de aptidão (fitness)

Utilizamos a seguinte função de fitness para avaliar nosso algoritmo genético

$$F(\text{indivíduo}) = 1 - \frac{\text{choqueProf} + \text{choqueSalas} + \text{disciplinasGem} + \text{indisProf}}{PCCP + PCCS + PCDG + PCIP} \quad (1)$$

na qual *indivíduo* representa uma possível solução para o quadro de horários de um curso/turno, *choqueProf* representa a quantidade de vezes em que os professores estão escalados para ministrar duas disciplinas diferentes em um mesmo horário, *choqueSalas* representa a quantidade de vezes em que salas são escaladas no mesmo horário, *disciplinasGem* representa a quantidade

de ocorrências de disciplinas geminadas, *indispProf* representa a quantidade de ocorrências de professores casados com suas respectivas informações de indisponibilidades.

Já *PCCP*, *PCCS*, *PCDG* e *PCIP* representam o pior caso para choque de professores, salas, disciplinas geminadas e indisponibilidade de professores para aquele curso/turno, respectivamente. O pior caso pode ser obtido através dos dados de entrada computando o somatório do mínimo entre a maior quantidade de slots,  $Q_{max}$ , em um semestre e a soma de todos os outros slots ocupados por um professor -  $Q_{max}$ .

Assim como em Alves et al. (2015), a razão entre as restrições violadas presentes em individuo e o pior caso das restrições, limitam o resultado de  $F(individuo)$  no intervalo  $[0,1]$ , tendo em vista que  $(choqueProf + choqueSalas + disciplinasGem + indispProf) \leq (PCCP + PCCS + PCDG + PCIP)$ .

### 3 SISTEMA WEB E API REST

#### 3.1 SISTEMA WEB

Como dito inicialmente para o desenvolvimento deste trabalho foram utilizadas apenas o framework Flask para o Backend e API. Para criação das interfaces de frontend foi utilizado o framework Vue JS. Foi desenvolvido um sistema web que abrange todos os passos necessários para a criação de um quadro de horários satisfatório para qualquer instituição de ensino. Neste capítulo tem como intuito demonstrar o funcionamento do sistema WEB como um todo e a aplicação da API Rest Relativa. O sistema foi construído de forma modular de forma que cada tarefa pertinente à geração de quadros de horários seja desenvolvida de forma simples e rápida.

Para a geração dos quadros se faz necessário o preenchimento prévio por parte da instituição de ensino ou das coordenações dos cursos informações sobre professores, suas indisponibilidades, eixos de ensino com seus respectivos cursos, disciplinas e salas. Além disso, ainda antes da geração dos quadros de horários, é necessário atribuir a cada disciplina uma sala de preferência e a cada professor um conjunto de disciplinas.

Esta seção trata de toda a funcionalidade dos módulos, incluindo capturas de telas e descrições mais detalhadas das funcionalidades desenvolvidas.

##### 3.1.1 Professores

O módulo que trata dos professores é composto de seções acessíveis por usuários com característica de administrador. Nesta seção estes usuários (que podem ser coordenadores, diretores de ensino, ou setores da instituição responsáveis) podem adicionar, editar e remover professores.

Figura 5 – Lista de professores

Timetable Home Sobre Olá, ALANI! [Logout](#)

👤 > Professor

Indisponibilidade Aleatória Indisponibilidade + Professor

Cód.	Nome Breve	CPF	E-mail	Créditos Livres/Totais	
1	ALAN / ALAN RABELO MARTIN	1	alanrabelo1@gmail.com	4/4	
2	PONCIANO / DIEGO PONCIANO DE OLIVEIRA LIMA	2	ponciano@gmail.com	4/4	
3	OTAVIO / OTAVIO ALCANTARA DE LIMA JUNIOR	3	otavio@gmail.com	4/4	
4	EURIPEDES / EURIPEDES CARVALHO DA SILVA	4	EURIPEDES@gmail.com	4/4	
5	RICARDO / FRANCISCO RICARDO NOGUEIRA	5	RICARDO@gmail.com	4/4	

29 professor(es) encontrado(s).

© 2018 500. Todos os direitos reservados.

Neste módulo, professores tem nome completo, nome abreviado, email, e CPF

Figura 6 – Tela de adicionar um novo professor

Timetable Home Sobre Olá, ALANI! [Logout](#)

👤 > Professor > Adicionar

Nome:

Nome breve:

CPF:

Créditos:

E-mail:

Administrador: ☐

© 2018 500. Todos os direitos reservados.

Cada professor individualmente também é responsável por adicionar possíveis indisponibilidades de horários, diretamente na sua tela de indisponibilidades.

Figura 7 – Adicionando indisponibilidade de horários do professor

Timetable Home Sobre Olá, ALANI! [Logout](#)

👤 > Professor > Indisponibilidade

Indisponibilidade

Segunda	Terça	Quarta	Quinta	Sexta
AB	AB	AB	AB	AB
CD	CD	CD	CD	CD
EF	EF	EF	EF	EF

Indisponibilidade

Segunda	Terça	Quarta	Quinta	Sexta
AB	AB	AB	AB	AB
CD	CD	CD	CD	CD
EF	EF	EF	EF	EF

Indisponibilidade

Segunda	Terça	Quarta	Quinta	Sexta
AB	AB	AB	AB	AB
CD	CD	CD	CD	CD
EF	EF	EF	EF	EF

© 2018 500. Todos os direitos reservados.

### 3.1.2 Eixos, cursos e disciplinas

Foi desenvolvido também o módulo do curso. Aqui os administradores podem criar, editar e excluir eixos, cursos e disciplinas. Eixos são uma maneira de dividir cursos por área, por exemplo, o eixo de Telemática, ou computação, que pode incluir disciplinas como o Bacharelado em Ciência da computação e o técnico em Redes, ou o Eixo Ambiental que pode ter alocado em si cursos como Engenharia Ambiental ou técnico em meio ambiente.

Figura 8 – Módulo Responsável por adicionar eixos, cursos e salas

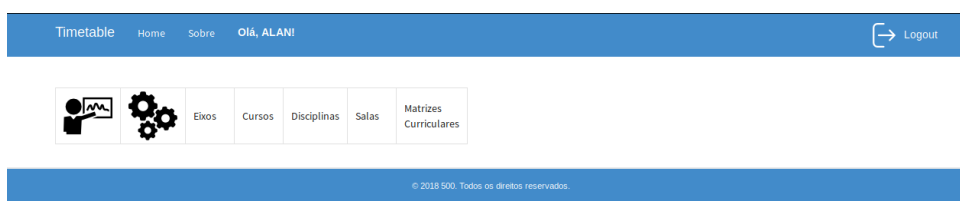


Figura 9 – Lista de Eixos

Timetable Home Sobre Olá, ALANI! Logout

Adicionar

Registro de Eixos

Descrição	Ativo	Remover	Editar
Telemática	true	Remover	Editar
Química	true	Remover	Editar
Ambiental	true	Remover	Editar

Há um total de 3 Eixo(s) cadastrado(s) no sistema.

© 2018 500. Todos os direitos reservados.

Figura 10 – Lista de Cursos

Timetable Home Sobre Olá, ALANI! Logout

Adicionar

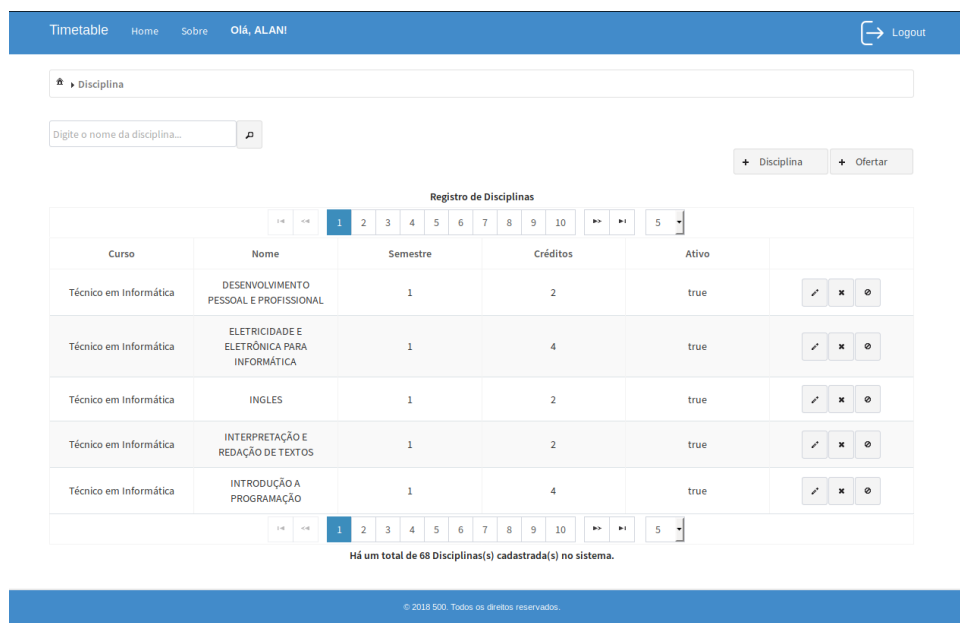
Registro de Cursos
















Nome	Ativo	Eixo	Remover	Editar	Períodos
Ciência da Computação	true	Telemática	Remover	Editar	Períodos
Técnico em Redes de Computadores	true	Telemática	Remover	Editar	Períodos
Engenharia Ambiental	true	Química	Remover	Editar	Períodos
Técnico em Meio Ambiente	true	Química	Remover	Editar	Períodos
Técnico em Automação Industrial	true	Ambiental	Remover	Editar	Períodos
Técnico em Informática	true	Telemática	Remover	Editar	Períodos
Minha computação	true	Telemática	Remover	Editar	Períodos

Há um total de 7 Curso(s) cadastrado(s) no sistema.

© 2018 500. Todos os direitos reservados.

Figura 11 – Lista de Disciplinas



Curso	Nome	Semestre	Créditos	Ativo	
Técnico em Informática	DESENVOLVIMENTO PESSOAL E PROFISSIONAL	1	2	true	  
Técnico em Informática	ELETRICIDADE E ELETRÔNICA PARA INFORMÁTICA	1	4	true	  
Técnico em Informática	INGLES	1	2	true	  
Técnico em Informática	INTERPRETAÇÃO E REDAÇÃO DE TEXTOS	1	2	true	  
Técnico em Informática	INTRODUÇÃO A PROGRAMAÇÃO	1	4	true	  

Dessa forma ao adicionar os diferentes eixos e cursos presentes na instituição o administrador inicia então o preenchimento das disciplinas relativas aos cursos. Para cada disciplina, é necessária a alocação de uma sala específica. Salas podem ser criadas, editadas e removidas nesse módulo. Ao atribuir uma sala a uma disciplina, o algoritmo vai então tratar essa restrição para que assim o conflito entre salas seja mínimo.

### 3.1.3 Ofertas

Depois que todas as entidades foram previamente cadastradas se faz necessário estabelecer uma ligação entre disciplinas e professores. Após uma prévia alocação ou acordo entre os professores sobre as disciplinas que cada um vai ministrar no semestre, essas ofertas são também cadastradas no sistema. Esse é o último passo necessário antes da execução do algoritmo.

## 3.2 GERAÇÃO DO QUADRO DE HORÁRIOS

### 3.2.1 Tela de Geração e Exibição

Após as informações inseridas na seção anterior podemos dar início à geração dos quadros de horário. Mesmo que o sistema WEB gere mais de uma solução ótima ele sempre vai exibir apenas um quadro que resolve o problema. Para tanto basta que o usuário acione o botão "Gerar quadro de horários" e aguarde. A geração do quadro demora entre 5 segundos e 12 minutos dependendo da quantidade de informações inseridas e da dificuldade de resolução do problema para a instituição e é exibido conforme as figuras 12 e ???. O quadro de horário

continua sendo gerado mesmo em *background*, dessa forma ao retornar ao sistema a solução gerada continua salva no banco de dados. Vale ressaltar também que imediatamente abaixo do quadro de horários de cada curso é exibida uma tabela exibindo dados da quantidade de choques de professores, salas, indisponibilidades feridas e disciplinas geminadas, dessa forma o usuário pode acompanhar a qualidade do quadro com relação às restrições impostas.

Figura 12 – Geração de Quadro de Horários

Timetable

Home

Sobre

OIA, ALANI

Logout

Horários

Adicionar

Período Letivo:

Gerar horário

HORÁRIO Ciência da Computação				
Segunda	Terça	Quarta	Quinta	Sexta
SEMESTRE 1 - Manhã				
....	FUNPROG - SALA 3 - Otavio	MATDIS - SALA 4 - Euripedes	INGINST - SALA 8 - Alisson	CAL 1 - SALA 1 - Alan
MATDIS - SALA 4 - Euripedes	CIR DIG - SALA 2 - Ponciano	CAL 1 - SALA 1 - Alan	FUNPROG - SALA 3 - Otavio	CIR DIG - SALA 2 - Ponciano
SEMESTRE 2 - Manhã				
LABPRO - SALA 9 - Carlos henrique	FIS 1 - SALA 7 - Shara	FIS 1 - SALA 7 - Shara	ALGLIN - SALA 5 - João claudio	....
CAL 2 - SALA 6 - Rodrigues	LABPRO - SALA 9 - Carlos henrique	ALGLIN - SALA 5 - João claudio	....	CAL 2 - SALA 6 - Rodrigues
SEMESTRE 3 - Manhã				
POO - LABORATÓRIO DE INFORMÁTICA 4 - Thiago queiroz	PLIN - LABORATÓRIO DE INFORMÁTICA 3 - David	PLIN - LABORATÓRIO DE INFORMÁTICA 3 - David	FIS 2 - LABORATÓRIO DE REDES 1 - Amauri	POO - LABORATÓRIO DE INFORMÁTICA 4 - Thiago queiroz
ARQCOMP - LABORATÓRIO DE INFORMÁTICA 1 - Cirineu	PEST - LABORATÓRIO DE INFORMÁTICA 2 - Jean	ARQCOMP - LABORATÓRIO DE INFORMÁTICA 1 - Cirineu	PEST - LABORATÓRIO DE INFORMÁTICA 2 - Jean	FIS 2 - LABORATÓRIO DE REDES 1 - Amauri
SEMESTRE 4 - Manhã				
SIS OPE - SALA 2 - Elder	COMDAD - LABORATÓRIO DE REDES 2 - Jonas	COMDAD - LABORATÓRIO DE REDES 2 - Jonas	SIS OPE - SALA 2 - Elder	CALNUM - LABORATÓRIO DE REDES SEM FIO - Ajalmar

Figura 13 – Lista exibindo a quantidade de choques

SIS OPE - SALA 2 - Elder	COMDAD - LABORATÓRIO DE REDES 2 - Jonas	LABORATÓRIO DE REDES 2 - Jonas	SIS OPE - SALA 2 - Elder	CALNUM - LABORATÓRIO DE REDES SEM FIO - Ajalmar
CALNUM - LABORATÓRIO DE REDES SEM FIO - Ajalmar	ESTDAD - LABORATÓRIO DE ELETRONICA E SISTEMAS EMBARCADOS - Aurenivia	BD - LABORATÓRIO DE REDES 1 - Gabriela	BD - LABORATÓRIO DE REDES 1 - Gabriela	ESTDAD - LABORATÓRIO DE ELETRONICA E SISTEMAS EMBARCADOS - Aurenivia
SEMESTRE 5 - Manhã				
IEECC - SALA 1 - Nivando	GRAFOS - LABORATÓRIO DE MÍDIAS DIGITAIS - Eugenio	ENG50F - LABORATÓRIO DE SISTEMAS DIGITAIS - Sandro	....	ANAALG - LABORATÓRIO DE VISÃO COMPUTACIONAL E INTELIGÊNCIA ARTIFICIAL - Wellington
ENG50F - LABORATÓRIO DE SISTEMAS DIGITAIS - Sandro	ANAALG - LABORATÓRIO DE VISÃO COMPUTACIONAL E INTELIGÊNCIA ARTIFICIAL - Wellington	GRAFOS - LABORATÓRIO DE MÍDIAS DIGITAIS - Eugenio	....	....
SEMESTRE 6 - Manhã				
LINPRO - SALA 4 - Corneli	LINPRO - SALA 4 - Corneli	....	APS - SALA 7 - Otavio	APS - SALA 7 - Otavio
METCIE - SALA 5 - Alan	....	REDCOMP 1 - SALA 6 - Ponciano	REDCOMP 1 - SALA 6 - Ponciano	METCIE - SALA 5 - Alan
SEMESTRE 7 - Manhã				
IA - SALA 3 - Fabiana	TEOCOMP - LABORATÓRIO DE INFORMÁTICA 1 - Jefferson	IA - SALA 3 - Fabiana	....	TEOCOMP - LABORATÓRIO DE INFORMÁTICA 1 - Jefferson
TCC 1 - SALA 9 - Ricardo	....	MICRO - SALA 8 - Euripedes	MICRO - SALA 8 - Euripedes	....
<div><div>Choque de professores0</div><div>Disciplinas geminadas0</div><div>Indisponibilidades feridas0</div><div>Choque de salas0</div><div>Avaliação [0 a 1]1,0</div></div>				



### 3.3 API REST

O protocolo Rest trabalha em conjunto com o protocolo HTTP, que é o protocolo utilizado na Web. Os principais métodos do HTTP, são os métodos GET, POST, PUT e DELETE. De acordo com Burke (2009) o funcionamento do HTTP é simples. Fazendo uso do modelo cliente-servidor, o cliente envia uma requisição ao servidor informando o método a ser utilizado, a localização do recurso a ser invocado, um conjunto de variáveis no cabeçalho e um corpo de mensagem opcional que pode ser qualquer formato tais como: HTML, XML ou JSON entre outros.

O REST é uma junção de vários estilos de arquitetura de rede, ou seja, é um estilo híbrido tais como Cliente/Servidor, Stateless, Interface Uniforme e Cache.

#### 3.3.1 Modelo cliente/servidor

O estilo Cliente/Servidor visa separar as responsabilidades e é muito utilizado em aplicações baseadas em rede, visa principalmente separar a interface do usuário do armazenamento de dados, facilitando assim a responsabilidade do servidor. No estilo cliente/servidor têm-se um Servidor responsável por oferecer um conjunto de serviços que serão invocados por aplicações clientes, o servidor disponibiliza seus recursos, para ser consumido por algum cliente independente da plataforma ou da linguagem escrita (FIELDING, 2000).

#### 3.3.2 Stateless

O estilo Stateless implica em não manter o estado entre a aplicação cliente e a aplicação servidora gerando assim um desacoplamento entre as aplicações, a cada nova requisição o cliente deve enviar todas as informações necessárias junto à requisição para que a mesma possa ser processada pelo servidor, isso diminui a sobrecarga dos servidores já que os mesmos não precisam armazenar o estado de cada cliente. O Estilo Stateless cita três propriedades importantes, são elas:

- Visibilidade: Cada pedido (requisição) contém todas as informações necessárias para que o servidor possa processá-lo.
- Confiabilidade: Permite a recuperação de falhas parciais.
- Escalabilidade: Por o servidor não precisar armazenar o estado de cada requisição o mesmo pode servir a mais pedidos em um tempo menor

### 3.3.3 Interface Uniforme

Estilo que define alguns princípios fundamentais como a identificação de recursos, diferentes representações, mensagens auto descritivas e recursos hipermídia (links)(FIELDING, 2000).

### 3.3.4 Cache

Estilo que permite fazer cache das informações para que não seja preciso o acesso ao banco de dados a todo o momento, é utilizado para otimização de desempenho do sistema (FIELDING, 2000).

## 3.4 RESTFUL

De acordo com Fielding (2000) Rest descreve os princípios que faz da Web uma imensa aplicação distribuída, oferecendo diversos serviços e aplicativos através de um conjunto de arquiteturas bem definidas. O termo Restful é utilizado para denominar aplicações que seguem os princípios Rest. O Rest obedece a alguns princípios que o fazer um serviço mundialmente utilizado para vários fins diferentes. Esses princípios são descritos a seguir.

### 3.4.1 Endereçamento

O princípio de endereçamento indica que todo recurso em um sistema deve poder ser acessado por meio de um identificador exclusivo (BURKE, 2009). Se faz necessário identificar os recursos a serem utilizados através de uma URL, tornando assim os recursos endereçáveis e possibilitando sua manipulação através dos métodos Http. Toda requisição Http deve ter uma URI.

Figura 14 – Exemplo de URI

`http://host:port/horarioaula/getTimetable`

Podemos dividir uma URI em alguns elementos. O primeiro elemento se refere ao protocolo utilizado para as comunicações, em nosso caso é o Http. O host:port conterá o endereço IP ou o endereço não resolvido, como `www.meuServidor.com` além da porta estabelecida para o acesso.

### 3.4.2 Métodos Http

Abaixo os métodos HTTP utilizados em sistemas Restful:

1. GET: Método utilizado para requisitar dados do servidor. É uma operação que não altera nada, apenas recebe informação.
2. PUT: Pode ser utilizado para criar ou atualizar dados, dessa forma solicitar que o servidor armazene os dados enviados.
3. DELETE: Método utilizado para remover um recurso.
4. POST: É geralmente utilizado para criar um recurso ou adicionar dados no servidor.
5. HEAD: Igual ao método GET, porém ao invés de retornar o response body retorna apenas o response code e os cabeçalhos associados à requisição.
6. OPTIONS: O método é utilizado para saber quais métodos podem ser utilizados em determinado recurso, ou seja, seu retorno sempre será um conjunto de métodos Http.

### 3.4.3 Stateless

"*Rest stateless* (sem estado) significa que não há dados de sessão de cliente armazenados no servidor. O servidor só registra e gerencia o estado dos recursos que ele expõe". (BURKE, 2009) Trabalhando de forma *stateless* o servidor não tem necessidade de salvar todos os dados dos clientes, conseguindo assim uma maior escalabilidade e evitando perda de recursos, tendo em vista que praticamente todos os custos dos procedimentos realizados são computacionais e não de armazenamento. O *stateless* define que em todas as requisições Http o cliente deve fornecer todos os dados necessários para a resposta do servidor, fazendo com que ele fique independente de respostas anteriores.

## 3.5 APLICAÇÃO

### 3.5.1 Azure

À medida que o número de empresas que migra seus produtos/serviços para a nuvem, cresce o número de opções de provedores de serviços em nuvem. Dentre as diversas opções que existem atualmente no mercado, uma que se mostra bem interessante é o Microsoft Azure. Fato reconhecido pela Gartner, que pelo quinto ano consecutivo elegeu o Azure como líder em Infraestrutura de Nuvem como Serviço (IaaS), Serviços de Armazenamento em Nuvem e

Plataforma de Aplicativos como Serviço (PaaS) (GARTNER, 2018). As vantagens oferecidas pelo Azure são muitas, com maior valor de destaque para:

- **Alta disponibilidade:** Ao contrário de outros provedores, a nuvem do Microsoft Azure oferece alta disponibilidade e redundância em data centers em escala global. Por esse motivo, o Azure pode oferecer um contrato de nível de serviço (SLA) de 99,95% (aproximadamente 4,38 horas de tempo de inatividade por ano), algo que a maioria das empresas não consegue atingir.
- **Segurança:** O Microsoft Azure tem um forte foco em segurança, seguindo o modelo de segurança padrão de Detectar, Avaliar, Diagnosticar, Estabilizar e Fechar. Juntamente com controles de segurança cibernética, este modelo permitiu que o Azure obtivesse várias certificações de conformidade, todas as quais estabelecem o Azure como líder em segurança de IaaS. Não apenas a plataforma é protegida, o usuário final também é coberto pelo Azure. Esse nível de proteção em vários níveis é essencial, pois as ameaças de segurança continuam se multiplicando diariamente no mundo todo, segmentando usuários finais e colocando em risco os dados da sua empresa. O Azure fornece serviços simples e fáceis de usar para maior proteção, como autenticação de vários fatores e requisitos de senha do aplicativo.
- **Escalabilidade:** O Microsoft Azure facilita a ampliação ou redução da capacidade de computação com nada mais que o clique de um botão. Com essa estrutura de escalabilidade, as empresas têm a flexibilidade de pagar apenas pelo que usam.
- **Custo por Eficácia:** Evidentemente, é essencial manter os orçamentos de TI em mente ao escolher um provedor de nuvem, e é por isso que a plataforma Microsoft Azure é tão atraente para muitas organizações. O preço pré-pago do Azure permite que as SMB gerenciem melhor seus orçamentos de TI, adquirindo apenas o quanto precisarem. Além disso, o ambiente de nuvem permite que as empresas iniciem aplicativos de clientes e aplicativos internos na nuvem, o que economiza custos de infraestrutura e reduz os encargos de hardware e manutenção no gerenciamento de TI interno.

Dentre as opções disponibilizadas pelo Azure para construir aplicações WEB, o escolhido foram os *Web Apps* pelas seguintes razões: (i) facilidade de implantação, além de fazê-lo com mais competência e velocidade que outros serviços em nuvem; (ii) capacidade de agregação de múltiplas aplicações (com o intuito de economizar recursos); (iii) auto-escalabilidade sem

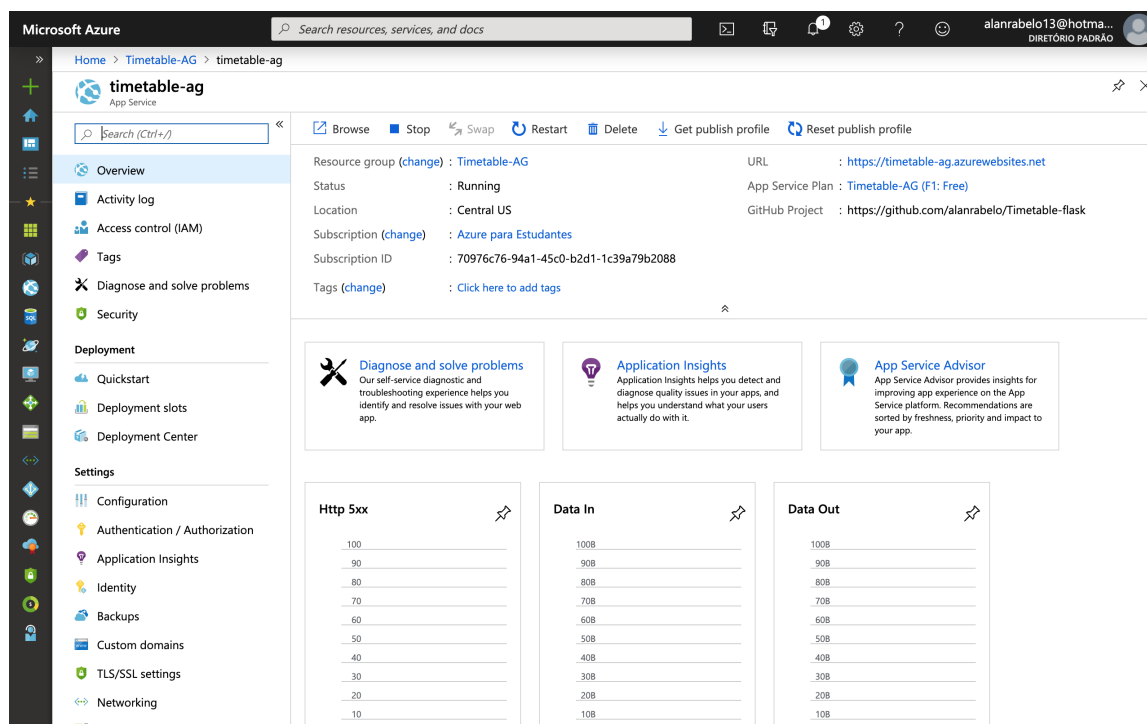


Figura 15 – Tela inicial de um Web app na azure

necessidade de reimplantação; e (iv) suporte para múltiplas linguagens de programação *open source*, tal como Python (muito utilizada atualmente para desenvolvimento de algoritmos de aprendizagem de máquinas e inteligência artificial). A figura 15 exhibe os detalhes de um Web App sendo executado na Azure incluindo sua URL, plano de serviço, localização geográfica e grupo de recursos.

O desenvolvimento de uma aplicação web pode ter sua produtividade aumentada quando utiliza-se *frameworks*. Isso porque tais ferramentas possuem muitas tarefas intrínsecas já prontas para uso, reduzindo assim o tempo de desenvolvimento bem como permitindo um foco maior na aplicação. Flask é a escolha para o projeto da API. Por ser um *microframework*, possui um *core* simples mas extensível. Por consequência, é uma escolha que nos permite trabalhar com mais aprofundamento em otimizações que são dependentes do problema.

Para o *deploy* da aplicação no ambiente em nuvem da Azure foi criada uma máquina virtual para testes que executa o código flask de forma remota.

No seção Deployment Center da Aplicação WEB no Azure é possível fazer o link de um repositório no Github, onde o projeto está hospedado. Desta forma qualquer *commit* na *branch* master no repositório executa uma *re-deploy* na Azure diretamente, permitindo assim que o Web App sempre esteja sincronizado com o desenvolvimento.

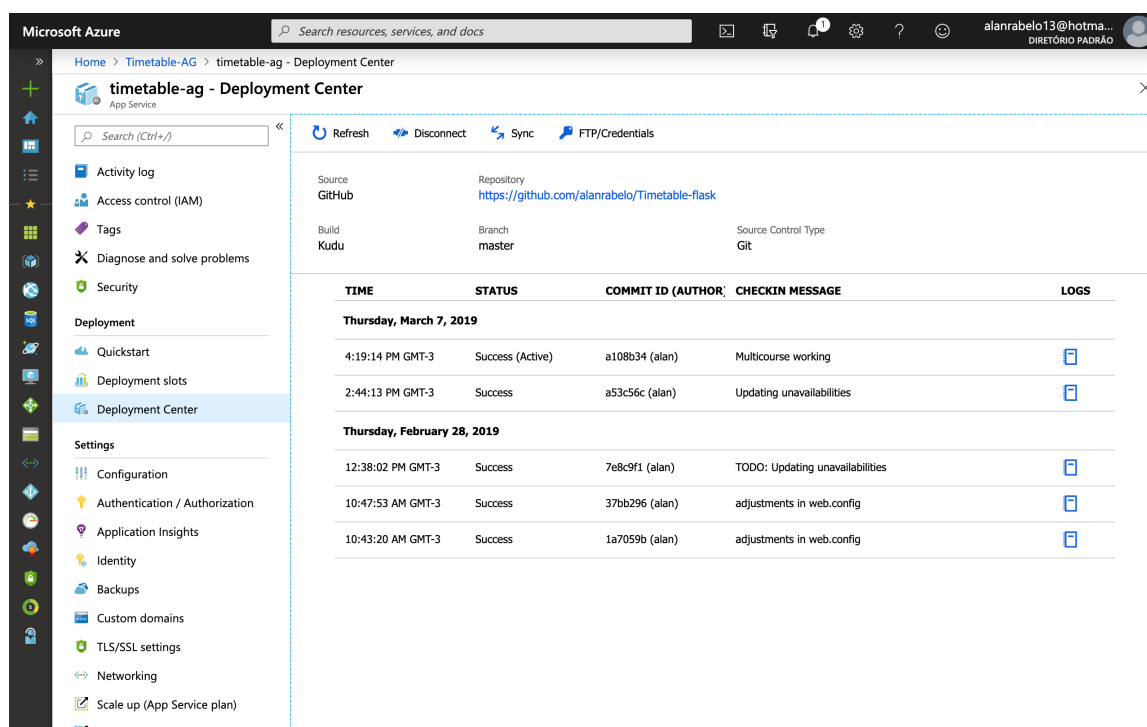


Figura 16 – Azure Deployment Center

A aplicação desenvolvida se utiliza apenas do método POST com os dados e informações necessárias sendo passadas no *body* e é completamente *stateless*, funcionando sem cache ou qualquer outro tipo de armazenamento de dados.

### 3.5.2 Entrada de dados

A aplicação recebe como entrada um arquivo JSON com informações sobre os cursos, turnos, disciplinas, salas, professores e suas indisponibilidades.

```

1  "cursos": {
2    "0": {
3      "name": "Ciência da Computação",
4      "semesters": 8
5    },
6    "1": {
7      "name": "Técnico em informática",
8      "semesters": 1
9    }
10 }

```

Listing 1: Cursos ofertados pela instituição

Na figura 1 é exibida a entrada de dados de cursos. Nela é criado um dicionário contendo apenas o nome do curso e o número máximo de semestres.

```
1  "turnos":{  
2    "0":{  
3      "name":"manhã"  
4    },  
5    "1":{  
6      "name":"tarde"  
7    }  
8  }
```

Listing 2: Possíveis turnos para alocação das disciplinas

Já na figura 2 é exibido um exemplo de entrada para os turnos. Bastando apenas passar objetos com um nome simples.

```
1  "semestres":{  
2    "0":{  
3      "name":"S1-Manhã",  
4      "turno":0  
5    },  
6    "1":{  
7      "name":"S2-Tarde",  
8      "turno":1  
9    }  
10 }
```

Listing 3: Descrição dos possíveis semestres com nome e id do turno previamente configurado

Já na figura 3 é exibido um exemplo de entrada para os possíveis semestres. Bastando apenas passar objetos com um nome simples para o semestre e seu turno.

```
1  "disciplinas":{
2    "0":{
3      "name":"Calc 1",
4      "semester":0
5    },
6    "1":{
7      "name":"Circuitos",
8      "semester":0
9    },
10   "2":{
11     "name":"FunProg",
12     "semester":0
13   },
14   "3":{
15     "name":"MatDisc",
16     "semester":0
17   },
18   "9":{
19     "name":"ARQcomp",
20     "semester":1
21   },
22   "10":{
23     "name":"Fisica 2",
24     "semester":1
25   },
26   "11":{
27     "name":"ProbEst",
28     "semester":1
29   }
30 }
```

Listing 4: Relação de todas as disciplinas com seu nome e respectivo semestre

A figura 4 exibe a relação de disciplinas possíveis, sendo necessário apenas o nome e o semestre em que é ofertada.



```
1  "professors":{
2    "0":{
3      "name":"Israel"
4    },
5    "1":{
6      "name":"Otávio"
7    },
8    "2":{
9      "name":"Saulo"
10   }
11 },
12 "indisponibilidades":{
13   "0":[
14     2,
15     5
16   ],
17   "1":[
18     3,
19     7
20   ],
21   "2":[
22     4
23   ]
24 }
```

Listing 5: Dados dos professores com suas indisponibilidades

A figura 5 exibe a relação de professores e suas respectivas indisponibilidades. Os professores possuem apenas um nome e suas indisponibilidades referenciadas por seus índices são baseadas no esquema encontrado no subseção 2.3.2 na tabela 1.

```
1  "salas":{
2    "0":{
3      "name":"Sala 0"
4    },
5    "1":{
6      "name":"Sala 1"
7    },
8    "2":{
9      "name":"Sala 2"
10   }
11 }
```

Listing 6: Descrição das salas

```
1  "ofertas":{
2    "0":{
3      "semester":0,
4      "disciplina":0,
5      "professor":0,
6      "creditos":2,
7      "sala":8,
8      "turno":0,
9      "curso":0
10   },
11   "1":{
12     "semester":0,
13     "disciplina":1,
14     "professor":1,
15     "creditos":2,
16     "sala":8,
17     "turno":0,
18     "curso":0
19   },
20   "2":{
21     "semester":0,
22     "disciplina":2,
23     "professor":2,
24     "creditos":2,
25     "sala":12,
26     "turno":0,
27     "curso":0
28   },
29   "3":{
30     "semester":0,
31     "disciplina":3,
32     "professor":3,
33     "creditos":2,
34     "sala":8,
35     "turno":0,
36     "curso":0
37   },
38   "4":{
39     "semester":1,
40     "disciplina":4,
41     "professor":7,
42     "creditos":2,
43     "sala":5,
44     "turno":0,
45     "curso":0
46   }
47 }
```

Listing 7: Descrição das ofertas de disciplinas

O conjunto de todos estes elementos deve ser passada como entrada do sistema como um único JSON.

### 3.5.3 Execução

A API utilizada gera uma URI específica para a requisição do tipo POST. A URI específica é `"/api/webservice"`. Dessa forma é encaminhada, a partir do cliente a requisição com o JSON definido na seção anterior, que é lido pela classe exibida na figura 17.

```

1  from Timetable.Timetable import Timetable
2  from flask import Flask, request, jsonify
3
4  app = Flask(__name__)
5
6  @app.route('/')
7  def home():
8      return 'Bem vindo à aplicação de quadros de horários!'
9
10
11 @app.route('/timetable', methods=['POST'])
12 def do_timetable():
13
14     input_as_json = request.json
15     timetable = Timetable()
16     return timetable.timetable_with_input(input_as_json)
17
18 if __name__ == '__main__':
19     app.run()
```

Figura 17 – Método responsável pela obtenção do JSON a partir da requisição recebida

A partir do momento que possuímos a variável `input_as_json`, já temos todos os dados suficientes para executar o algoritmo genético, que possui uma classe responsável pelo processamento chamada `Timetable` responsável por executar a função `timetable_with_input(input_as_json)`. Os parâmetros do GA são definidos previamente pela própria classe `Timetable`. O resultado da execução dessa função é um arquivo JSON contendo a relação de cursos, semestres e horários preenchidos com a melhor organização possível de disciplinas.

Um exemplo da resposta fornecida pelo sistema pode ser visualizada na figura 18. Nela podemos encontrar dados dos cursos e seus respectivos semestres, além de turnos, manhã, tarde e noite e seus horários (AB, CD ou EF) preenchidos de acordo com o resultado do algoritmo genético. Com o JSON organizado desta forma o solicitante da API pode facilmente exibir de

```

1 {
2   "Computacao": {
3     "S1": {
4       "manhã": {
5         "segunda": {
6           "AB": {
7             "disciplina": { "id": 1, "nome": "Cálculo 1" },
8             "professor": { "id": 1, "nome": "Pedro" }
9           },
10          "CD": { "disciplina": { "id": 2, "nome": "RNA" },
11                "professor": { "id": 2, "nome": "Ajalmar" }
12          }
13        },
14        "terça": {
15          "AB": { "disciplina": { "id": 3, "nome": "MatDis" },
16                "professor": { "id": 1, "nome": "Pedro" }
17          },
18          "CD": { "disciplina": { "id": 4, "nome": "IA" },
19                "professor": { "id": 2, "nome": "Ajalmar" }
20          }
21        }
22      }
23    }
24  }
25 }

```

Figura 18 – JSON retornado pelo sistema

forma gráfica e salvar ou imprimir o resultado utilizando seus sistemas finais, como um sistema Web, ou no sistema acadêmico da própria instituição.

#### 4 Resultados da aplicação

Para avaliar o algoritmo, utilizamos, como estudo de caso, os cursos de nível técnico e superior do IFCE - Campus Maracanaú - da área de Computação. A melhor configuração encontrada por Alves et al. (2015), que se utiliza de valores de Crossover 50%, População 75 e Mutação 1/25. O *hardware* utilizado foi: iMac late 2017 com um processador Intel Core i3 com 3.6GHz de frequência de operação, 16GB de RAM, SSD com capacidade de 240GB. O sistema operacional utilizado foi o Mac OS Mojave.

Foram utilizados dois ambientes para a avaliação. No simples não se consideraram as restrições de horário de professores, enquanto que no complexo, nos três cursos foram adicionadas 4 *slots* de indisponibilidades em slots de tempo completamente aleatórias para cada professor.

Foram avaliadas 50 execuções da mesma configuração de algoritmo para cada ambiente. Os resultados demonstram um tempo satisfatório: Para o ambiente 1, tivemos uma média de aproximadamente 4,7 segundos, enquanto que no ambiente 2 obtivemos uma média de aproximadamente 2,15 minutos.

Tabela 2 – Resultados dos testes para o ambiente que considera indisponibilidades: Taxa de mutação, crossover, tamanho da população, média de gerações, número de execuções com valor de fitness igual a 1 e o tempo médio de execução de cada configuração.

Testes	Mutação (%)	Crossover (%)	População	$\mu$ Gerações	fitness = 1	$\mu$ Runtime (sec)
1	1/12	35	75	359	6/10	98.4
2	1/12	35	200	125	9/10	20.2
3	1/12	35	600	38	10/10	25.3
4	1/12	50	75	280	6/10	110
5	1/12	50	200	210	9/10	18.5
6	1/12	50	600	52	5/10	17.4
7	1/40	50	75	630	6/10	76.3
8	1/40	50	200	118	7/10	18.3
9	1/40	50	600	65	10/10	24.5
10	1/25	50	75	35	9/10	156.1
11	1/25	50	200	75	8/10	34.0
12	1/25	50	600	43	6/10	32.7
13	1/25	60	75	240	7/10	103.3
14	1/25	60	200	146	8/10	49.7
15	1/25	60	600	48	6/10	38.7

Tabela 3 – Comparação com trabalhos relacionados

Parâmetro	(Shara, 2015)	Este Trabalho
Turmas	16	23
Professores	21	35
Salas	não se aplica	15
População	75	75
Reprodução	50% OX	50% OX
Mutação	1/25	1/25
Seleção	Aleatória	Aleatória
Elitismo	-	-
Gerações necessárias	493	553
Tempo de Execução	3,7 min	1,56 min
Fitness Final [0, 1]	1	1

A partir da tabela 2 pudemos verificar a configuração ideal para nosso algoritmo. Utilizando esta configuração, comparamos os resultados com o trabalho anterior de (ALVES et al., 2015). A partir desta comparação podemos observar, na tabela 3 que nosso algoritmo necessita de um tempo menor, com um pouco mais de gerações, tendo em vista que ele opera sobre um problema mais complexo e ainda conta com a adição das salas de aula, ou seja, é uma restrição a mais que torna o algoritmo um pouco mais lento, mas continuando eficiente com valor de função de fitness máxima.

## 5 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foram realizados o estudo e a aplicação de algoritmos genéticos ao problema do quadro de horários multi-curso, a partir da necessidade de um Web Service e uma API Rest que atenda a necessidade de geração destes quadros para o IFCE Campus Maracanaú e para os demais campus e instituições de ensino que, porventura, necessitem deste tipo de aplicação.

Dessa forma, neste trabalho foram desenvolvidos sistemas que solucionam o problema de forma ótima, reduzindo consideravelmente o tempo gasto pela instituição para criação e otimização manual do mesmo. Vale ressaltar que o Web Service pode ser acessado a partir de uma URI pública, através de algum tipo de autenticação, tornando-se um serviço passível de ser utilizado por toda a comunidade acadêmica, sem que a mesma se preocupe com detalhes de implementação de AG, ou de aquisição de um sistema web completo, podendo, caso opte simplesmente pela API, preocupar-se apenas com a interface que venha por ventura ser utilizado pelos coordenadores de cursos ou funcionários da instituição para inserir e converter dados para o formato exigido na api.

Como trabalhos futuros poderia ser acrescentada uma restrição que evitasse que professores que ensinam em mais de uma sede da mesma instituição tenham aulas em duas ou mais sedes diferentes num mesmo dia. Desta forma um professor que leciona uma disciplina do Mestrado em Ciência da Computação em Fortaleza, não necessitaria se deslocar até Maracanaú para ensinar uma disciplina do Bacharelado, evitando assim custos financeiros e de tempo de deslocamento para professores.

Além disso um modelo poderia ser pensado para favorecer a escolha dos alunos levando em conta horários desejados e evitando choque de horários ou disciplinas para os mesmos.

## REFERÊNCIAS

- ABOUELHAMAYED, A. F.; MAHMOUD, A. S.; SHAABAN, T. T.; SALAMA, C.; YOUSEF, A. H. An enhanced genetic algorithm-based timetabling system with incremental changes. In: **2016 11th International Conference on Computer Engineering Systems (ICCES)**. [S.l.: s.n.], 2016. p. 122–127.
- ALVES, S. S. A.; OLIVEIRA, S. A. F.; NETO, A. R. R. A novel educational timetabling solution through recursive genetic algorithms. In: **2015 Latin America Congress on Computational Intelligence (LA-CCI)**. [S.l.: s.n.], 2015. p. 1–6.
- BURKE, B. **RESTful Java with JAX-RS**. First. [S.l.]: O'Reilly, 2009.
- FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. [S.l.: s.n.], 2000.
- GARTNER. **Gartner's Magic Quadrant for Infrastructure as a Service**. 2018. <<https://azure.microsoft.com/pt-br/resources/gartner-iaas-magic-quadrant/en-us/>>. Online; accessed 12 July 2018.
- GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.
- MITCHELL, M. **An Introduction to Genetic Algorithms (Complex Adaptive Systems)**. [S.l.]: The MIT Press, 1996. ISBN 0262133164.
- MOUSA, H. M.; EL-SISI, A. B. Design and implementation of course timetabling system based on genetic algorithm. In: **2013 8th International Conference on Computer Engineering Systems (ICCES)**. [S.l.: s.n.], 2013. p. 167–171.
- RODRIGUEZ, N.; MARTINEZ, J.; FLORES, J. J.; GRAFF, M. Solving a scholar timetabling problem using a genetic algorithm - study case: Instituto tecnologico de zitacuaro. In: **2014 13th Mexican International Conference on Artificial Intelligence**. [S.l.: s.n.], 2014. p. 197–202.
- SMITH, D. A. K. A.; DUKE, D. Hopfield neural networks for timetabling: formula- tions, methods, and comparative results. In: **Comput. Ind. Eng.** [S.l.: s.n.], 2003.