

Uma ferramenta flexível para a geração automática de horários de cursos de graduação baseada em algoritmos genéticos*

Osvaldo Airon B. Cavalcanti Filho¹, Marcus Carvalho²

¹Departamento de Ciencias Exatas (DCX) – Universidade Federal da Paraíba (UFPB)
Cep 58297-000 – Rio Tinto – PB – Brasil

{osvaldo.airon,marcuswac}@dcx.ufpb.br

Abstract. *Elaborating class schedules for undergraduate courses in a complex task, because it involves many restrictions that affects logistics of teachers, students and classrooms. Some schedule solutions are not viable, for example when different classes of the same teacher are allocated at the same time, while other solutions are viable but have undesirable class times. The amount of possible class schedules is generally very large, which makes it a very difficult job to perform manually with not very clear criteria. Therefore, this work presents a tool to automatically generate class schedules for undergraduate courses based on genetic algorithms. Four criteria have already being developed to find the best solution, but it is also flexible to allow the inclusion of new criteria and customize the weights of each criteria according to the needs of each course. The evaluation was performed by generating class schedules for the Information Systems and Computer Science courses of the Federal University of Paraíba, campus IV, which rapidly converged to a viable and optimal solution.*

Resumo. *Elaborar quadros de horários de cursos de graduação é uma tarefa complexa, pois envolve diversas restrições que podem afetar a logística de professores, alunos e salas de aula. Algumas soluções de horários podem ser inviáveis, por exemplo quando aulas diferentes de um mesmo professor são alocadas no mesmo horário, enquanto outras são viáveis mas possui horários indesejáveis. A quantidade de combinações possíveis de alocação de horários geralmente é muito grande, o que torna o trabalho manual extremamente difícil, com critérios pouco objetivos e difícil de serem satisfeitos. Diante disto, este trabalho apresenta uma ferramenta para a geração automática de horários de cursos de graduação baseada em algoritmos genéticos. A ferramenta já possui quatro critérios desenvolvidos para a escolha da melhor solução, mas é flexível para permitir a inclusão de novos critérios e a customização dos pesos de cada critério de acordo com a necessidade do curso. A avaliação foi realizada a partir da geração de horários para os cursos de Sistemas de Informação e Licenciatura em Ciência da Computação da Universidade Federal da Paraíba, campus IV, convergindo rapidamente para uma solução viável e ótima.*

*Trabalho de conclusão de curso, sob orientação do professor Marcus Carvalho submetido ao Curso de Bacharelado em Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAEE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO.

1. Introdução

Elaborar o horário das aulas de cursos de graduação é uma das atividades administrativas mais importantes que ocorrem em todas as instituições acadêmicas e também um dos maiores desafios, tendo em vista que gerar horários viáveis e adequados precisa-se levar em consideração uma série de possibilidades e restrições que devem ser atendidas, muitas vezes conflitantes entre si.

Horários mal elaborados acarretam em uma série de problemas tanto para alunos quanto para professores. Algumas soluções de horários podem ser inviáveis, por exemplo quando aulas diferentes de um mesmo professor ou de disciplinas de um mesmo período são alocadas no mesmo horário. Outros problemas comumente encontrados são: (1) aulas da mesma disciplina alocadas no mesmo dia, sobrecarregando alunos e professores e tornando as aulas improdutivas; (2) fragmentação de horários de aulas de um mesmo período, por exemplo com uma aula no início da manhã e a próxima só no fim da tarde, prejudicando a logística de estudo, trabalho e deslocamento de alunos e professores.

A quantidade de combinações possíveis de alocação de horários geralmente é muito grande, o que torna o trabalho manual extremamente difícil, com critérios pouco objetivos e difícil de serem satisfeitos. O processo de alocação de horário escolar é um problema de otimização combinatória no qual é mais fácil verificar se o resultado apresentado está correto do que obter analiticamente tal solução em tempo polinomial. Esse processo é denominado de *timetabling* escolar, que devido à sua complexidade computacional e a dificuldade da aplicação de algoritmos exatos, normalmente aplicam-se técnicas heurísticas ou probabilísticas. Uma abordagem heurística utilizada para resolução deste tipo de problema é a técnica de otimização combinatória denominada de algoritmos genéticos, que tem apresentado bons resultados de acordo com a literatura [Barboza, 2003].

Diante disto, este trabalho tem como objetivo principal a criação de uma ferramenta flexível para a geração de horários de cursos de graduação, avaliando a sua viabilidade e eficiência de acordo com critérios objetivos implementados com base nas necessidades do corpo docente e discente. Os objetivos específicos deste trabalho são:

- Modelar o problema e a solução usando algoritmos genéticos.
- Implementar critérios para a seleção do melhor horário e uma função de custo que considere pesos diferentes para cada critério.
- Desenvolver uma ferramenta flexível que permita a inclusão de novos critérios e a configuração dos pesos de cada critério na função de custo.

Este trabalho está estruturado da seguinte forma: a seção 2 descreve a fundamentação teórica com os conceitos da abordagem dos algoritmos genéticos. A seção 3 apresenta os trabalhos relacionados. A seção 4 descreve a metodologia aplicada para a resolução do problema e para a implementação da ferramenta. A seção 5 traz a avaliação, com análises e discussões dos resultados obtidos e, finalmente, a seção 6 apresenta a conclusão do trabalho.

2. Fundamentação Teórica

Algoritmos Genéticos (AGs) fazem parte de um grupo conhecido como Algoritmos Evolutivos e são baseados em um processo coletivo de aprendizagem dentro de uma

população de indivíduos, cada um dos quais representa um ponto no espaço de busca de soluções para um dado problema [Ribeiro Filho, 2000]. Os AGs foram desenvolvidos por Holland [1975] ao explorar os processos de adaptação de sistemas naturais e suas possíveis aplicabilidades em projetos de software. Este tipo de algoritmo faz o uso de métodos presentes na seleção natural como mutação, cruzamento de espécies, hereditariedade, seleção natural e recombinação da seleção natural. O princípio básico do funcionamento dos AGs se apoia no critério da seleção dos indivíduos sobreviventes a cada geração e que após sucessivas gerações deve gerar uma população de indivíduos cada vez mais aptos [Leite et al., 2006].

Durante a existência dos indivíduos, processos como mutação e cruzamento atuam sobre o material genético armazenado nos cromossomos, sendo o conteúdo do cromossomo um conjunto de genes. Tais processos promovem uma variabilidade genética dos seres vivos na população. A Tabela 1 a seguir ilustra uma comparação entre os termos usados na biologia e o seu respectivo em AG.

Tabela 1. Comparativo entre termos da Biologia e AGs

Termo	Biologia	AGs
Cromossomo	É o conjunto completo de genes do organismo	Estrutura de dados que codifica uma solução para o problema
Gene	Unidade de hereditariedade e que controla as características do organismo	Parâmetro codificado no cromossomo, elemento ou variável da estrutura representada pelo cromossomo
Fitness	Valor que indica o grau de aptidão do indivíduo	Fornece para cada indivíduo a medida de quão próximo a uma solução considerada satisfatória ele se encontra
Indivíduo	Integrante da população	Composto por um conjunto de genes (Cromossomo) e um valor de fitness

A Figura 1 exemplifica o funcionamento de um algoritmo genético. O primeiro passo é a definição e criação da população inicial; nesta etapa, é definido o conjunto de indivíduos que fazem parte da população. Em seguida, todos os indivíduos são submetidos a um cálculo de aptidão que avalia o seu desempenho de acordo com as características desejadas, este processo por sua vez é denominado de “*fitness*”. A função de avaliação mede a qualidade de um cromossomo como solução do problema em questão. É a forma de diferenciar as boas das más soluções para um problema. Seguindo essa lógica de diferenciação entre as boas e más soluções, a função aplica um sistema de bonificação aos indivíduos que se adequam melhor aos requisitos estabelecidos e uma punição aos que não possuam as características desejáveis. Este processo de bonificação e punição baseado em critérios é baseado em dois tipos: restrições rígidas (*hard constraints*), que são consideradas críticas pois, uma vez que violadas, tornam a solução inviável; e restrições leves (*soft constraints*), que por sua vez quando violadas não tornam a solução inviável, mas são indesejáveis e devem ser contabilizadas durante a avaliação e seleção do melhor indivíduo [Linden, 2006].



Figura 1. Exemplo gráfico do fluxo algoritmo genético.
Fonte: [Lacerda, 1999]

Após o cálculo de aptidão de todos os indivíduos da população, é verificado se o critério de parada foi satisfeito, ou seja, se a quantidade de gerações já foi suficiente e a execução pode ser encerrada. Caso não seja satisfeito, é realizado o processo da seleção dos indivíduos que se tornam elegíveis para o processo de criação da população da geração seguinte. Um dos processos para criar novos indivíduos é denominado cruzamento (*crossover*), no qual cada novo indivíduo possui uma mistura das características de seus pais. Realizado o processo de cruzamento, o novo indivíduo gerado sofre uma alteração em um ou mais dos seus genes, em um processo denominado mutação (*mutation*), que ocorre a fim de causar no indivíduo uma maior variabilidade genética.

As operações de *fitness*, *crossover* e *mutation* irão ocorrer até que o conjunto de indivíduos sejam capazes de criar uma nova população. O processo se repete até que a condição de parada seja atendida e quando atendida é selecionado o melhor ou N melhores indivíduos que compõem aquela população, com base nas suas aptidões.

3. Trabalhos Relacionados

Esta seção discute alguns trabalhos relacionados ao problema proposto no artigo.

Reis Filho and Origuela [2018] propuseram uma ferramenta de otimização de quadros escolares na qual foi aplicada no Centro Universitário Unifafibe. A ferramenta por sua vez não permitia a configuração de parâmetros do algoritmo genético e apresentava complicações ao inserir os dados cadastrais referentes a disciplinas, cursos e períodos. A ferramenta ainda não fazia diferenciação entre restrições rígidas e leves.

Freitas et al. [2007] desenvolveram a ferramenta Kairós, aplicada na Faculdade Ruy Barbosa (FRB) que teve como base em sua implementação o conceito de algoritmo genéticos. Com o intuito de elaborar quadros de horários dentro do contexto da FRB, a ferramenta apresentou uma média de 1 a 3 conflitos na grade curricular e um tempo de processamento em minutos de 35 a 78 minutos com populações variando entre 3000 à 5000 e o número de gerações 500 à 1000 respectivamente.

Em comparação a estes trabalhos descritos, a solução proposta aqui neste artigo

teve como diferenciais:

- Fácil importação dos dados cadastrais.
- Adição e customização de restrições.
- Diferenciação entre restrições rígidas e leves.
- Tempo de processamento menor que a ferramenta Kairós.
- Análise de violações por restrições ao fim da execução.
- Gráfico com acompanhamento de valores do fitness.

4. Metodologia

Para a resolução do problema, este trabalho propõe o desenvolvimento de uma ferramenta flexível baseada em algoritmos genéticos. Um dos diferenciais desta ferramenta em relação aos trabalhos relacionados é a capacidade de extensão das funções de restrição, uma vez que para adicioná-las se faz necessário alterar apenas um módulo do sistema. Além disso, a ferramenta permite ao usuário configurar a execução do algoritmo genético e seus operadores, customizar os pesos das restrições e, ao final da execução, exportar os resultados obtidos em cada análise juntamente com o quadro de horários da melhor solução encontrada. Esta flexibilidade torna a ferramenta capaz de se adaptar a diferentes contextos e realidades de diferentes cursos. O código fonte da ferramenta é aberto para a comunidade e encontra-se no Github¹, permitindo a liberdade de personalização da mesma.

4.1. Concepção de uma ferramenta extensível

Para a concepção da ferramenta, foi utilizada a linguagem de programação Python, por ela já possuir bibliotecas especializadas na construção de algoritmos genéticos e estrutura de dados, além de ser uma linguagem facilmente legível e extensível. Esta ferramenta admite dois papéis de usuários, o usuário desenvolvedor que faz a implementação de novas restrições bem como seus pesos e o usuário final que faz a execução do algoritmo como as restrições implementadas e os operadores configurados. A biblioteca escolhida para realizar o processo de criação dos indivíduos e operações genéticas foi a DEAP² (Distributed Evolutionary Algorithms in Python), que é uma estrutura de computação evolutiva que procura tornar os algoritmos explícitos e as estruturas de dados transparentes. Ela apresenta uma série de funcionalidades que permitem aplicar os processos evolutivos (vistos na Figura 1) de maneira compreensível. A arquitetura do DEAP é composta por duas estruturas simples, o *criador* (*creator*) e a caixa de ferramentas (*toolbox*), sendo o pacote de criação uma meta-fábrica que permite a criação de entidades em tempo de execução por herança e composição. As informações relevantes para cada entidade como atributos ou funções podem ser adicionados dinamicamente. A caixa de ferramentas permite ao usuário desenvolvedor customizar e registrar os operadores e as funções do seu algoritmo evolucionário. Por exemplo, se o usuário final precisar utilizar um operador de cruzamento, o mesmo poderá escolher o mais adequado para a resolução do problema dentre os vários já implementados.

¹Disponível em: <https://github.com/lbblou/Gerador-de-Horarios>

²Disponível em: <https://deap.readthedocs.io/en/master> Acesso em: 30/jun/2020

4.2. Representação do cromossomo

No processo de criação das entidades do sistema, optou-se pelo uso do paradigma de orientação a objetos para definir as entidades criadas e os seus relacionamentos. Foram criadas três entidades: disciplinas, aulas e horários. Estas entidades são acessadas de forma externa pelo algoritmo genético. A Figura 2 ilustra o modelo de entidade relacionamento que compõem as entidades.

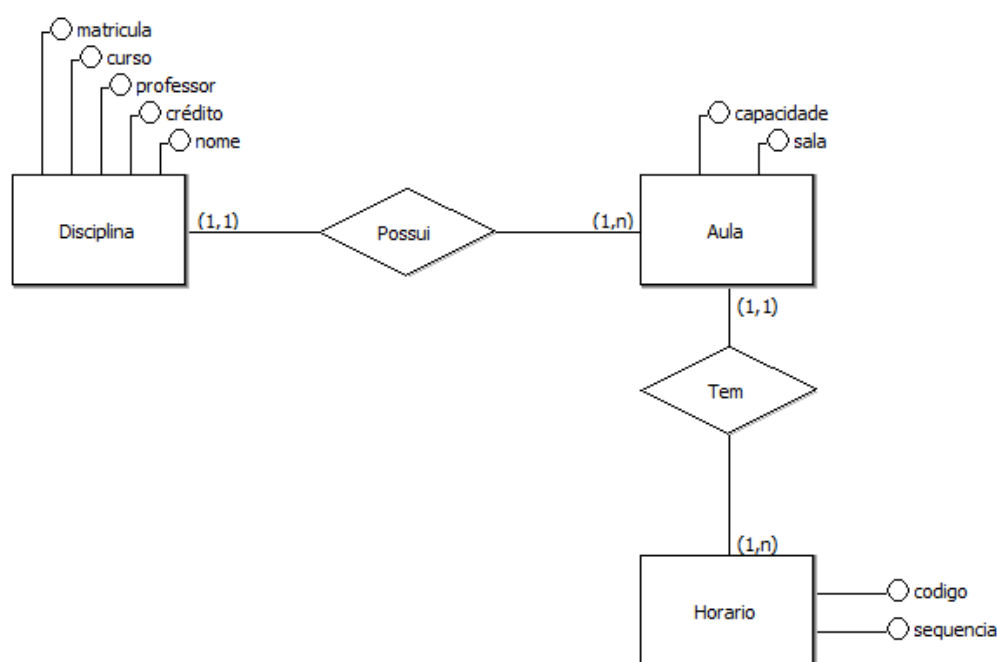


Figura 2. Modelo entidade relacionamento para definição das entidades do sistema. Fonte: Autor

4.3. Dados de entrada e configuração da ferramenta

O fornecimento dos dados para a criação da população inicial e execução do algoritmo se dá através de passagem de argumentos para a ferramenta. A mesma admite 3 arquivos de entrada no formato CSV³:

- *disciplinas.csv*: cada linha representa uma disciplina ofertada e compõem a entidade Disciplina do sistema. Suas colunas são: nome da disciplina, matrícula, professor, curso e créditos.
- *horarios.csv*: cada linha representa um horário disponível para alocação e compõem a entidade Horário do sistema. Suas colunas são: código do horário e sequência.
- *salas.csv*: cada linha representa uma sala disponível para alocação e compõe a entidade Sala do sistema. Suas colunas são: código da sala e capacidade.

³Disponível em: <https://tools.ietf.org/html/rfc4180> Acesso em: 25/jul/2020

A ferramenta possui um outro método de importação dos dados de entrada a partir de arquivos tipo JSON⁴. Utilizando a biblioteca CherryPy⁵ foi possível configurar e criar um servidor minimalista dentro da aplicação e assim criar 3 endpoints para cada arquivo especificado, sendo eles *disciplinas.json*, *horarios.json* e *salas.json* permitindo a integração da ferramenta com outros sistemas web.

4.4. Representação do indivíduo

No modelo do indivíduo, que representa uma solução de horário, foi criado um array de inteiros no qual cada elemento representa um índice de um horário, gerado a partir de uma enumeração de horários disponíveis, e cada índice do array representa um horário alocado para uma disciplina. Então temos que o tamanho do array é $N * \bar{x}$ onde N é a quantidade de disciplinas e $\bar{x}$ a quantidade de aulas por disciplina sendo $N * \bar{x}$ a quantidade total de aulas, cada aula com seu horário respectivo sendo este um gene nesta representação e o indivíduo o conjunto de horários. A Figura 3 mostra um exemplo de indivíduo. Ao total são mapeados neste exemplo 110 de genes, este valor corresponde a quantidade de disciplinas que foram cadastradas multiplicado pela quantidade de aulas para cada disciplina. Para a resolução deste problema, foram inseridas 55 disciplinas e a quantidade de aulas por disciplina foram 2.

```
[19, 9, 3, 2, 13, 12, 12, 11, 11, 7, 1, 3, 4, 4, 1, 9, 6, 13, 8, 18, 19, 5,
9, 14, 18, 2, 6, 0, 3, 19, 3, 15, 18, 6, 19, 6, 18, 10, 15, 11, 10, 4, 0, 6,
2, 14, 15, 14, 19, 11, 2, 8, 9, 4, 7, 13, 13, 14, 10, 19, 15, 8, 0, 4, 2, 5,
0, 10, 13, 1, 1, 13, 0, 6, 10, 12, 15, 11, 7, 11, 7, 3, 7, 5, 6, 19, 8, 2,
9, 17, 5, 18, 4, 15, 1, 0, 2, 11, 0, 14, 13, 4, 4, 17, 12, 8, 17, 16, 14, 5]
```

Figura 3. Exemplo de indivíduo e seus respectivos genes.

Fonte: Autor

A quantidade de horários disponíveis configurados que compõem a enumeração de horários para a resolução do problema foram 20 horários. A seguir é ilustrado a enumeração de horários e os valores configurados.

```
1 from enum import Enum
2 class HorarioE(Enum):
3     SEGUNDA_8H=1
4     SEGUNDA_10H=2
5     SEGUNDA_13H=3
6     SEGUNDA_15H=4
7     TERCA_8H=5
8     TERCA_10H=6
9     TERCA_13H=7
10    TERCA_15H=8
11    QUARTA_8H=9
12    QUARTA_10H=10
13    QUARTA_13H=11
```

⁴Disponível em: <https://www.json.org/json-en.html>. Acesso em: 25/jul/2020

⁵Disponível em: <https://cherrypy.org> Acesso em: 20/jun/2020

```

14     QUARTA_15H=12
15     QUINTA_8H=13
16     QUINTA_10H=14
17     QUINTA_13H=15
18     QUINTA_15H=16
19     SEXTA_8H=17
20     SEXTA_10H=18
21     SEXTA_13H=19
22     SEXTA_15H=20

```

Listagem 1. Conjunto de horários configurados na enumeração.

Cada horário configurado teve um intervalo de 2 horas podendo ainda ser customizado para o intervalo desejado.

4.5. População inicial

Após a leitura dos arquivos de entrada, o algoritmo gera a população inicial contendo uma lista de D objetos do tipo *Disciplina* da seguinte forma: com base na quantidade de créditos informada na entidade *Disciplina*, calcula-se a quantidade x de aulas que cada disciplina terá, de tal forma que para cada disciplina são instanciadas x aulas contendo um objeto do tipo *horário*. O cromossomo é composto por $N = D * \bar{x}$ genes, onde $\bar{x}$ é a média de aulas por disciplina. Cada gene representa um horário associado a uma aula que faz parte de uma disciplina. O código de cada horário é atribuído através de uma distribuição uniforme com valores $h \in [0, H - 1]$, onde H é a quantidade de horários disponíveis e h é o índice de um horário no array de horários disponíveis. Os valores gerados para h na população inicial serão sorteados aleatoriamente permitindo repetição. Tem-se então os indivíduos da população inicial composto por N genes, sendo N a quantidade de aulas semanais para todas as disciplinas.

4.6. Função de aptidão e a customização de restrições

Durante o planejamento dos critérios e das funções de restrições que seriam relevantes para o quadro de horário em um contexto geral, foram definidas restrições que poderiam ser classificadas como restrições rígidas ou suaves. Nesta versão da ferramenta, com base nas necessidades dos horários dos cursos relacionados a este trabalho, foram implementadas as seguintes restrições iniciais:

- **R1:** disciplinas do mesmo curso e período não podem ter aulas no mesmo horário.
- **R2:** disciplinas ministradas pelo mesmo professor não podem ter aulas no mesmo horário.
- **R3:** uma disciplina não pode ter mais de duas aulas no mesmo dia.
- **R4:** disciplinas que tenham aulas no horário de 15 horas serão penalizadas.

As restrições R1 e R2 são do tipo rígidas, ambas com pesos de 1000. E as restrições R3 e R4 do tipo suave ambas com pesos de 10.

Pensando em criar uma ferramenta que permitisse a fácil configuração das funções de restrições foi discutido como o usuário desenvolvedor da ferramenta irá fazer a importação de novas funções. Inicialmente o usuário desenvolvedor da ferramenta irá fazer a customização de novas restrições em apenas um módulo da ferramenta denominado de *ativacoes.py*, este módulo é responsável por todas as funções de restrições no

qual a função de aptidão iria acessá-las e obter as respectivas violações. No decorrer da implementação e na integração deste módulo com o pacote de ferramentas da biblioteca DEAP, percebeu-se que seria necessário definir para cada restrição o seu respectivo peso. A Listagem 2 apresenta o registro dos pesos de cada função de restrição e o acesso dessas funções pela função de aptidão.

```

1 pesos = [1000,1000,10,10]
2 toolbox=tools.Toolbox()
3 creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
4 creator.create("Individual", list, fitness=creator.FitnessMin)
5 def fitness_function(individual):
6     dList = GeradorObject.get_list_horarios_by_enum()
7     disp = GeradorObject.recreateDisciplines(disciplines,dList,
8         individual,QUANTIDADE_AULAS_POR_DISCIPLINA)
9     ativacoes = []
10    ativacoes.append(R1(disp))
11    ativacoes.append(R2(disp))
12    ativacoes.append(R3(disp))
13    ativacoes.append(R4(disp,dList))
14    result = 0
15    for i in range(len(ativacoes)):
16        result += ativacoes[i].get('violations') * pesos[i]
17    return result,

```

Listagem 2. Definição da função de aptidão

Inicialmente é criado uma lista de pesos onde cada índice corresponde ao peso de uma função de restrição, em seguida é definido qual tipo de função de aptidão vai ser utilizada pelo algoritmo juntamente com os pesos definidos, o pacote de criação do DEAP instância uma classe denominada *FitnessMin*. A mesma por sua vez é passada como atributo para o objeto *Individual* registrado pela caixa de ferramentas. O *FitnessMin* é como iremos avaliar a solução, esta função tem como objetivo principal minimizar os valores do fitness, isto ocorre porque nas funções de restrições é avaliado a quantidade de violações que a mesma teve. Então diz-se que o indivíduo que tiver a menor quantidade de violações será a melhor solução daquela geração. Conforme visto na Listagem 2 a *fitness_function* é responsável por resgatar as violações de cada restrição dentro do vetor de *ativacoes*, estes valores são multiplicados pelo respectivo peso de cada ativação, por fim é realizado um somatório destes valores e o DEAP se responsabiliza por fazer a aplicação do resultado para aquela solução.

4.6.1. Atribuição de novos horários

Conforme mencionado na seção 4.2, a representação da solução ocorre através de um array de genes composto por número inteiros, sendo cada um dos genes, índices da enumeração de horários. Foi implementado um algoritmo que faz o processo de manipulação dos objetos criados a partir do array de genes. O algoritmo criado recebe na sua função de manipulação alguns parâmetros, sendo estes: array de genes, enumeração horários e por fim a lista de disciplinas criadas. O processo de manipulação de objetos é feito da seguinte maneira: o array de genes é fragmentado em sub-vetores, cada sub-vetor é do tamanho da quantidade de aulas por semana $\bar{x}$. Cada índice do array de genes

agora fragmentado é correspondente ao índice de uma disciplina da lista de disciplinas. A Figura 4 exemplifica um array de genes fragmentado.

```
[[19, 9], [6, 7], [0, 6], [11, 11], [8, 14], [1, 13], [16, 10], [15, 14], [7, 13], [12, 6], [6, 2], [14, 13], [11, 16], [4, 0], [9, 9], [1, 8], [0, 2], [11, 14], [13, 13], [9, 10], [7, 10], [8, 14], [17, 4], [12, 7], [0, 10], [17, 13], [11, 9], [3, 11], [18, 8], [8, 11], [17, 15], [9, 11], [12, 1], [13, 18], [2, 9], [4, 2], [2, 18], [18, 15], [16, 1], [1, 19], [13, 15], [3, 4], [3, 3], [9, 19], [2, 15], [6, 1], [0, 14], [17, 4], [7, 14], [0, 8], [16, 8], [15, 12], [4, 4], [13, 8]]
```

Figura 4. Fragmentação do array de genes.

Fonte: Autor

Após o processo de fragmentação é feita uma iteração percorrendo a lista de disciplinas na qual são recuperados os códigos de horários de cada disciplina. Durante este processo é feita a atribuição do novo valor do código para o horário iterado. A atribuição é feita a partir dos elementos de cada sub-vetor. Por exemplo: se determinado que $\bar{x}$ tem o valor de duas aulas por semana, então a disciplina do índice zero irá conter duas aulas com seus horários respectivos e o sub-vetor com índice zero terá dois elementos que são os índices da lista de enumeração. Em seguida a partir destes índices é recuperado o código do horário e por fim realizada a atribuição do novo valor.

4.7. Seleção dos indivíduos e elitismo

Para a seleção dos indivíduos, foi escolhido o sistema de torneio juntamente com a técnica do elitismo, que consiste em pegar uma fração mínima dos melhores indivíduos e transferi-los para a próxima geração garantindo assim uma qualidade na solução. A biblioteca DEAP não implementa esta técnica explicitamente, então foi necessário sobrescrever o método de seleção para manter uma fração dos melhores indivíduos. A Listagem 3 ilustra a codificação da função de seleção juntamente com a técnica do elitismo.

```
1 toolbox.register("mutate", tools.mutUniformInt, low=0, up=GeradorObject.
   get_len_horarios_enumeration()-1, indpb=0.10)
2
3     def selElitistAndTournament(individuals, k, frac_elitist, tournsize
   ):
4         return tools.selBest(individuals, int(k*frac_elitist)) + tools.
   selTournament(individuals, int(k*(1-frac_elitist)), tournsize=
   tournsize)
```

Listagem 3. Função de seleção de indivíduos aplicando seleção por torneio e elitismo.

No registro da função de seleção é passado como parâmetro a fração de indivíduos que sofrerão o elitismo. Na Listagem 3 o valor passado é de 10 por cento, os demais indivíduos restantes da amostra irão participar do torneio no qual será escolhido o melhor indivíduo daquela amostra. O sistema de torneio consiste em pegar uma amostra da população e selecionar dois ou mais indivíduos aleatórios comparando-se suas aptidões sendo o mais apto escolhido. A operação irá se repetir até que a quantidade de indivíduos atingida for igual ao tamanho da população.

4.8. Cruzamento

Para a realização do processo de cruzamento, foram implementados dois sistemas de recombinação e o usuário final da ferramenta poderá escolher qual utilizar. O primeiro foi o de *recombinação em um corte*, onde é escolhido um ponto aleatório do array de genes de dois indivíduos e uma das metades desses cromossomos trocam de posição entre os indivíduos gerando assim dois novos indivíduos filhos. A Figura 5 exemplifica o processo de cruzamento por recombinação em um corte.

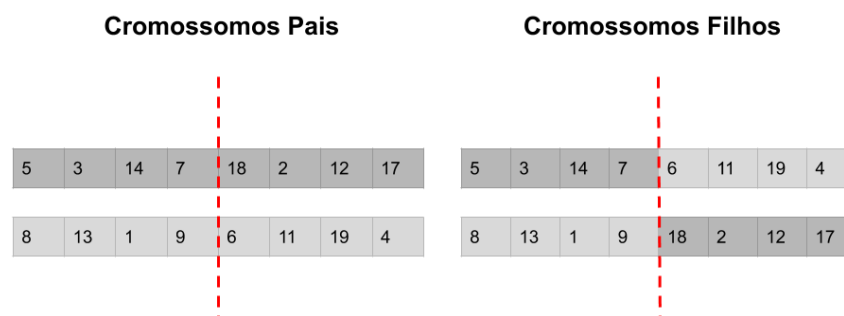


Figura 5. Exemplo de cruzamento de um corte.

O segundo foi o de *recombinação por dois cortes*, onde é escolhido dois pontos aleatórios no array de genes de dois indivíduos e tudo que está entre os pontos de corte é trocado gerando assim dois novos indivíduos filhos. A Figura 6 exemplifica o processo de cruzamento por recombinação de dois cortes.

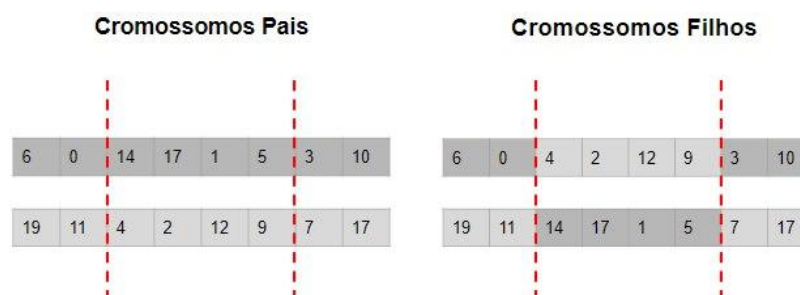


Figura 6. Exemplo de cruzamento de dois cortes.

Embora fique a critério do usuário final escolher por qual sistema de cruzamento utilizar, os dois sistemas de cruzamento não tiveram divergências no que se refere ao tempo de execução do algoritmo, contudo vale ressaltar que os indivíduos que tinham o cruzamento de dois cortes configurados apresentaram um grau de aptidão menor e um número menor de violações por restrições suaves, se comparado com os indivíduos que possuíam o sistema de cruzamento por um corte.

4.9. Mutação

Para o processo de mutação foi escolhida a função *MultiUniformInt* do DEAP. Esta função recebe como parâmetros: o array de genes, o valor mínimo, o valor máximo e a taxa de mutação definida em um número entre 0 e 1. O valor mínimo e máximo são referentes à quantidade de elementos na enumeração de horários. Durante o processo de mutação, um gene aleatório é escolhido e tem o seu valor alterado pela função *MultiUniformInt* o novo valor inserido é condizente ao intervalo de números definidos no valor mínimo e máximo. A taxa de mutação pode ser configurada pelo usuário da ferramenta, contudo se não for definida o algoritmo atribui uma taxa padrão de 10 por cento.

4.10. Configuração da execução do algoritmo genético

A configuração e execução do algoritmo genético pode ser feita através da interface de linha de comando, passando argumentos que são processados utilizando o módulo *argparse*⁶, que é um facilitador para a criação de interfaces de linha de comando. O programa define quais argumentos serão criados e o *argparse* faz o processo de identificar estes argumentos e como tratá-los. Este módulo ainda conta com mensagens de ajuda e emite erros quando o usuário final fornece argumentos inválidos ao programa, conforme observado na Figura 7.

```
usage: main.py [-h] [-i I] [-g G] [-m M] [-c C]

Configuração de execução do algoritmo genético

optional arguments:
  -h, --help            show this help message and exit
  -i I                  Define a quantidade de indivíduos, número inteiro
  -g G                  Define o número de gerações, número inteiro
  -m M                  Define a taxa de mutação em porcentagem Ex: 0.10
  -c C                  Define os tipos de cruzamento Digite(1) para cruzamento de um corte ou (2) para cruzamento de dois cortes, número inteiro

digite python3 main.py --man para visualizar o manual. Digite python3 main.py -h para visualizar configurações do algoritmo genético.
```

Figura 7. Configuração da execução do algoritmo pela linha de comando.

A ferramenta ainda possui um pacote utilitário que tem como objetivo auxiliar o usuário final, seja na execução do algoritmo ou no processo de importação dos arquivos. Estas informações estão presentes no manual da ferramenta executando o argumento *python3 main.py --man*.

4.11. Critério de Parada

Nesta versão inicial da ferramenta, foi estabelecido que o critério de parada seria determinado pelo número de gerações configurados. Então se configurado que na execução do algoritmo o número de gerações fosse 500, independentemente do valor do grau de aptidão do indivíduo, o critério de parada só será satisfeito ao fim da iteração pelo número de gerações.

5. Avaliação

Na avaliação da ferramenta, foi realizada uma bateria de testes com algumas configurações iniciais. O primeiro cenário de teste foi com os seguintes parâmetros:

⁶Disponível em: <https://docs.python.org/3/library/argparse.html> Acesso em: 26/jun/2020

- quantidade de indivíduos: 500
- número máximo de gerações: 500
- taxa de mutação: 20%
- tipo de cruzamento: um corte

Neste cenário de teste, a ferramenta encontrou duas restrições que foram violadas: a restrição R1 e R4 citadas na seção 4.5. O tempo para execução do algoritmo genético foi de aproximadamente 10 minutos. O segundo cenário de teste realizado teve como parâmetros:

- quantidade de indivíduos: 1000
- número máximo de gerações: 500
- taxa de mutação: 20%
- tipo de cruzamento: dois cortes

Ao final da execução, a ferramenta encontrou apenas uma restrição violada (R4), sendo a quantidade de violações para essa restrição igual a 1. Este valor identifica que entre todas as aulas da solução apenas uma disciplina possui uma alocada no horário das 15 horas. Por fim, o tempo de execução para este segundo caso de teste foi de aproximadamente 21 minutos. Todas essas informações são exportadas pela ferramenta em um arquivo de texto como ilustrado a seguir na Figura 8.

```
Quantidade De Indivíduos: 1000
Quantidade De Gerações:500
Taxa De Mutação:0.20
Cruzamento: Cruzamento dois cortes
Tempo de Execução (Minutos): 21
Quantidade de restrições violadas: 1
Restrição índices:numero de violações: (3: violações: 1)
```

Figura 8. Arquivo com informações da execução do algoritmo.

O terceiro cenário de teste realizado teve como parâmetros:

- quantidade de indivíduos: 3000
- número máximo de gerações: 500
- taxa de mutação: 20%
- tipo de cruzamento: dois cortes

Neste último cenário, nenhuma das restrições foram violadas e o algoritmo conseguiu atingir a solução ótima para as restrições configuradas. O tempo de execução do algoritmo foi de aproximadamente 63 minutos.

Complementando esta análise foi possível observar também o acompanhamento de valores da função de aptidão para o melhor indivíduo de cada geração. Quanto menor o valor do fitness do indivíduo, melhor é aquela solução. O gráfico ilustrado na Figura 9 exibe a minimização do fitness do indivíduo à cada geração.

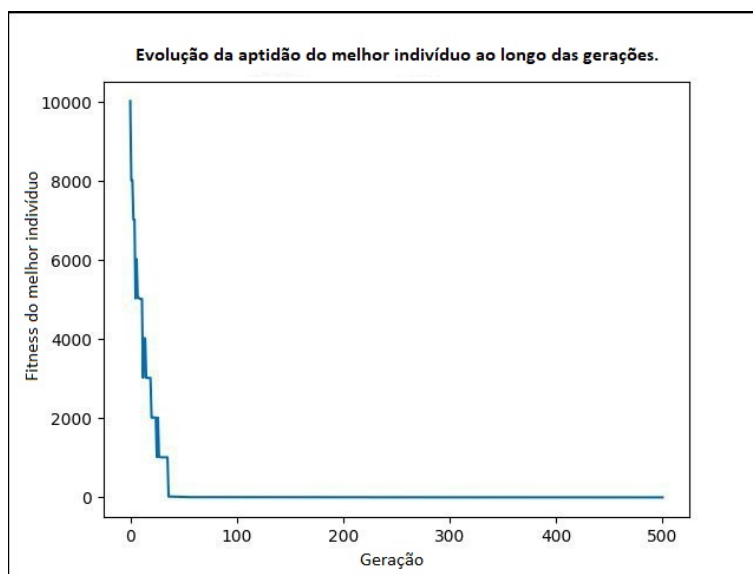


Figura 9. Evolução da aptidão do melhor indivíduo ao longo das gerações.

A solução encontrada não apresentou nenhum conflito de horários, validando as restrições R1 e R2 consideradas rígidas. Ao fim desta análise, pode-se perceber que a medida em que a quantidade de indivíduos aumenta, o valor de aptidão do indivíduo tende a diminuir gerando assim uma solução melhor à cada iteração das gerações.

6. Conclusão

Este trabalho teve como objetivo implementar uma ferramenta capaz de criar quadro de horários de maneira simples e eficaz utilizando o conceito de algoritmos genéticos. A ferramenta desenvolvida propôs uma nova forma de configuração de critérios tornando a extensível e preparada para se adaptar a outros contextos. Ela também é capaz de informar ao usuário qual ou quais restrições estão sendo violadas dada as configurações de execução do algoritmo, tornando assim os quadros de horários submetidos a avaliação mais confiáveis no que se diz respeito às diretrizes impostas pelos professores durante o planejamento das mesmas. A ferramenta possui ainda recursos para importação dos dados de entrada suportando arquivos do tipo .csv e .json.

Com base nos resultados obtidos na bateria de testes foi possível concluir ainda que a ferramenta oferece à exportação de arquivos ilustrando as violações encontradas e os valores da função de fitness, como também o tempo de execução do algoritmo. Nos (Apêndices A e B) são ilustrados os quadros de horários gerados a partir do último cenário de teste onde nenhuma das restrições configuradas foram violadas.

Como possíveis trabalhos futuros, pode-se criar uma interface web que se comunique com a ferramenta através dos endpoints criados para importação dos arquivos e implementar recursos para execução do algoritmo genético através de novos endpoints, passando as configurações de entrada do algoritmo genético sendo elas: número de indivíduos, quantidade de gerações, taxa de mutação e o tipo de cruzamento. A entidade Aula não foi utilizada como parâmetro das funções de restrições implementadas, apenas como um acesso para os horários. Em uma versão futura, pode-se aprimorar funções de restrições que faça um melhor aproveitamento desta modelagem. O critério de parada

mencionado anteriormente pode ainda ser customizado de maneira que quando um indivíduo chegasse a um grau de aptidão desejado a execução do algoritmo fosse finalizada.

Referências

- FJR Barboza. Resumos de artigos sobre algoritmos genéticos. 32f. *Mestrado em Mecatrônica-Escola de Engenharia, UFBA, Salvador*, 2003.
- Cherze C Freitas, Priscilla RB Guimarães, Manoel CM Neto, and Frederico JR Barboza. Uma ferramenta baseada em algoritmos genéticos para a geração de tabela de horário escolar. *SÉTIMA ESCOLA REGIONAL DE COMPUTAÇÃO Bahia-Sergipe. Vitória da Conquista:[sn]*, 2007.
- JH Holland. *Adaptation in natural and artificial systems*, university of michigan press, ann arbor. 1975.
- EG Lacerda. M; carvalho, acplf 1999 introdução aos algoritmos genéticos. *Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais*, pages 99–150, 1999.
- Patricia Teixeira Leite, Adriano AFM Carneiro, and André CPLF Carvalho. Aplicação de algoritmos genéticos na determinação da operação ótima de sistemas hidrotérmicos de potência. *SBA: Controle & Automação Sociedade Brasileira de Automática*, 17(1): 81–88, 2006.
- Ricardo Linden. *Algoritmos genéticos: uma importante ferramenta da inteligência computacional*. Brasport, 2006.
- José Humberto Nunes dos Reis Filho and Leandro C Origuela. Otimização de quadros horários escolares por meio de uma ferramenta baseada em algoritmos inteligentes. 2018.
- Geraldo Ribeiro Filho. Melhoramentos do algoritmo genético construtivo e novas aplicações em problemas de agrupamento. *INPE, São José dos Campos*, 2000.

A. Apêndice I - Melhor quadro de horário para o curso de Licenciatura em Ciência da Computação

1º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08.00 -10.00	Calculo 1 Josevandro	Introdução a Programção TUR 2 Ana Liz	Introdução a Programção TUR 2 Ana Liz	Metodologia Thaise	IP Vanessa/Rafael
10.00- 12.00	Matematica Elementar Laudelino		Matematica Elementar Laudelino	Introdução a Programação TUR 1 Ayla	Introdução a Programação TUR 1 Ayla
13.00-15.00	Metodologia Thaise	Introdução ao Computador Scaico	IP Vanessa/Rafael	Calculo 1 Josevandro	Introdução ao Computador Scaico
15.00-17.00					
2º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08.00 -10.00	Ling. de Programação Ayla	Vetorial Wendhel		Arquitetura I Augusto	
10.00- 12.00		Arquitetura I Augusto	Ling. de Programação Ayla	Vetorial Wendhel	
13.00-15.00			Calculo II Josevandro		Calculo II Josevandro
15.00-17.00					

3º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08.00 -10.00	POO Rodrigo Vilar	Arquitetura II Rafael	Álgebra Linear Wendhel		Estr. de Dados Wagner
10.00- 12.00		Álgebra Linear Wendhel		POO Rodrigo Vilar	Arquitetura II Rafael
13.00-15.00		Álgebra Linear e Geom. Analitica Wendhel	Álgebra Linear e Geom. Analitica Wendhel	Estr. de Dados Wagner	
15.00-17.00					
4º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08.00 -10.00	Ling. Declarativas Joelson			SO Raquel	Pesquisa Aplicada Pasqueline
10.00- 12.00	Pesquisa Aplicada Pasqueline		SO Raquel	Mod. de dados Pasqueline	
13.00-15.00		Ling. Declarativas Joelson			Mod. de dados Pasqueline
15.00-17.00					

5º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08:00 -10:00				Banco de Dados Vanessa	Empreend. DSCA
10:00- 12:00			Banco de Dados Vanessa		Redes Augusto
13:00-15:00		Empreend. DSCA	Redes Augusto	Teoria da Comp. Joelson	Teoria da Comp. Joelson
15:00-17:00					
6º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08:00 -10:00		Estatística Joseilme		Análise e Projeto Rodrigo Vilar	
10:00- 12:00			Análise e Projeto Rodrigo Vilar		Sistemas Multimídia Thaise
13:00-15:00				Sistemas Multimídia Thaise	Estatística Joseilme
15:00-17:00					

7º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08:00 -10:00	Intelig. Artificial e Sistemas Espec. Marcus	ESII Thaise	EAD Flávia		
10:00- 12:00	ESII Thaise	Eng. de Software Leandro	Intelig. Artificial e Sistemas Espec. Marcus	Eng. de Software Leandro	
13:00-15:00		EAD Flávia	PMI Flávia		PMI Flávia
15:00-17:00					

B. Apêndice II - Melhor quadro de horário para o curso de Sistemas da Informação

1º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08:00 -10:00	Introdução ao Computador <i>Ana Liz</i>	Metodologia Científica <i>Substituto</i>	Metodologia Científica <i>Substituto</i>	Matemática Elementar <i>Laudelino</i>	Administração I <i>Eliane</i>
10:00- 12:00	Administração I <i>Eliane</i>	Introdução à Programação TUR 2 <i>Ana Liz</i>			Matemática Elementar <i>Laudelino</i>
13:00-15:00	Introdução à Programação TUR 2 <i>Ana Liz</i>		Introdução ao Computador <i>Ana Liz</i>	Introdução à Programação TUR 1 <i>Ayla</i>	Introdução à Programação TUR 1 <i>Ayla</i>
15:00-17:00					
3º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08:00 -10:00		Álgebra Linear e Geom. Analítica <i>Wendhel</i>		Alg. e Estrutura de Dados 1 <i>Wagner</i>	
10:00- 12:00			Álgebra Linear e Geom. Analítica <i>Wendhel</i>		
13:00-15:00		Alg. e Estrutura de Dados 1 <i>Wagner</i>		Prog. Orientada a Objetos - POO <i>Raquel</i>	Prog. Orientada a Objetos - POO <i>Raquel</i>
15:00-17:00					

5º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08:00-10:00	Redes de Computadores <i>Hacks</i>	Banco de Dados I <i>Leandro</i>	Sistemas Operacionais <i>Raquel</i>	Eng. de Software Aplicada <i>Juliana/Yuska</i>	
10:00-12:00		Eng. de Software Aplicada <i>Juliana/Yuska</i>	Banco de Dados I <i>Leandro</i>	Pesquisa Aplicada <i>Adson</i>	Sistemas Operacionais <i>Raquel</i>
13:00-15:00	Pesquisa Aplicada <i>Adson</i>		Redes de Computadores <i>Hacks</i>		
15:00-17:00					
7º período	[2] SEG	[3] TER	[4] QUA	[5] QUI	[6] SEX
08:00-10:00		Sistemas Distribuídos <i>Marcus</i>	Sistemas de Apoio a Gestão <i>Luiz</i>		Desenv. de Sist. Corporativos <i>Leandro</i>
10:00-12:00	Desenv. de Sist. Corporativos <i>Leandro</i>	Intelig. Artificial e Sistemas Espec. <i>Marcus</i>	Auditoria e Seg. de Sistemas <i>Hacks</i>	Sistemas Distribuídos <i>Marcus</i>	Auditoria e Seg. de Sistemas <i>Hacks</i>
13:00-15:00	Gestão da Qualidade SW <i>Yuska</i>	Gestão da Qualidade SW <i>Yuska</i>	Intelig. Artificial e Sistemas Espec. <i>Marcus</i>	Sistemas de Apoio a Gestão <i>Luiz</i>	
15:00-17:00					