Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Santa Helena
Bacharelado em Ciência da Computação

TRABALHO DE FUNDAMENTOS DE SISTEMAS INTELIGENTES

O USO DO ALGORITMO GENÉTICO PARA MONTAR UMA GRADE DE HORÁRIOS

Nathan S. Ribeiro Guimarães [1]

SANTA HELENA
06/2024

[1]nathan.2019@alunos.utfpr.edu.br

INTRODUÇÃO

A geração de grades de horários é um problema complexo enfrentado por instituições educacionais devido às múltiplas restrições envolvidas, como a disponibilidade de salas, professores e horários específicos para cada disciplina. A otimização dessa tarefa visa maximizar a utilização eficiente dos recursos enquanto minimiza conflitos de horários, tanto para professores quanto para alunos.

1. MATERIAL E MÉTODOS

Para fazer a criação do algoritmo genético usamos o Google Colab, ou *Collaboratory*, é uma plataforma gratuita baseada na nuvem oferecida pelo Google. Ela fornece um ambiente de notebook interativo e colaborativo que permite a criação e execução de código diretamente no navegador, sem a necessidade de configurar ou instalar qualquer software no seu computador.

1.1. MATERIAL

1.1.1. Colab

O Colab é um serviço gratuito do Jupyter Notebook hospedado que não requer configuração para uso e oferece acesso gratuito a recursos de computação, incluindo GPUs e TPUs. O Colab é adequado principalmente para aprendizado de máquina, ciência de dados e educação.

1.1.2. Algoritmo Genético

O pioneiro no desenvolvimento de algoritmos genéticos foi John Henry Holland. Ele introduziu os conceitos fundamentais dos algoritmos genéticos, que são métodos de otimização e busca inspirados na evolução natural. Holland foi um matemático e cientista da computação americano que fez contribuições significativas para a teoria dos sistemas adaptativos complexos.

Em 1975, ele publicou o livro *"Adaptation in Natural and Artificial Systems"*, que estabeleceu as bases para os algoritmos genéticos como uma abordagem inovadora na computação evolutiva. Este trabalho marcou um ponto de virada no uso de simulações evolutivas na programação e otimização

Esses algoritmos são amplamente utilizados em diversas áreas, incluindo otimização, aprendizado de máquina e inteligência artificial, devido à sua capacidade de encontrar soluções aproximadas para problemas complexos.

O funcionamento básico de um AG pode ser descrito em várias etapas principais:

1. **Inicialização**: Criação de uma população inicial de soluções (indivíduos), geralmente de forma aleatória.
2. **Avaliação**: Cada indivíduo é avaliado por uma função de aptidão (*fitness function*), que determina a qualidade da solução representada pelo indivíduo.
3. **Seleção**: Indivíduos são selecionados para reprodução com base na sua aptidão. Métodos comuns incluem a roleta, torneio e seleção por ranking.
4. **Cruzamento (*Crossover*)**: Combinação de pares de indivíduos selecionados para gerar novos indivíduos (filhos), herdando características dos pais.
5. **Mutação**: Aplicação de pequenas alterações aleatórias nos indivíduos gerados para manter a diversidade genética na população.
6. **Substituição**: A nova geração de indivíduos substitui parte ou toda a população atual.
7. **Iteração**: O processo é repetido até que um critério de parada seja atingido, como um número máximo de gerações ou uma solução com aptidão satisfatória.

Formulação Matemática

A formulação matemática de um AG envolve a representação das soluções e a definição das operações genéticas. Um exemplo básico de um AG pode ser descrito da seguinte forma:

1. **Representação das Soluções**:

$$P(t) = \{x_1(t), x_2(t), \ldots, x_N(t)\}$$

Onde **P(t)** é a população no tempo **t**, e **xi(t)** representa um indivíduo na população.

2. **Função de Aptidão**:

$$f : \mathbb{R}^n \to \mathbb{R}$$

A função de aptidão **f** avalia a qualidade das soluções.

3. **Seleção**:

$$p_i = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

Onde **pi** é a probabilidade de seleção do indivíduo **xi**.

4. **Cruzamento**:

$$\begin{cases} \text{Se rand}() < p_c, & \text{realizar cruzamento} \\ \text{Caso contrário,} & \text{manter os pais} \end{cases}$$

Onde **pc** é a probabilidade de cruzamento.

5. **Mutação**:

$$\begin{cases} \text{Se rand}() < p_m, & \text{aplicar mutação} \\ \text{Caso contrário,} & \text{manter o indivíduo} \end{cases}$$

Onde **pm** é a probabilidade de mutação. Dessa forma temos um processo semelhante a essa imagem:

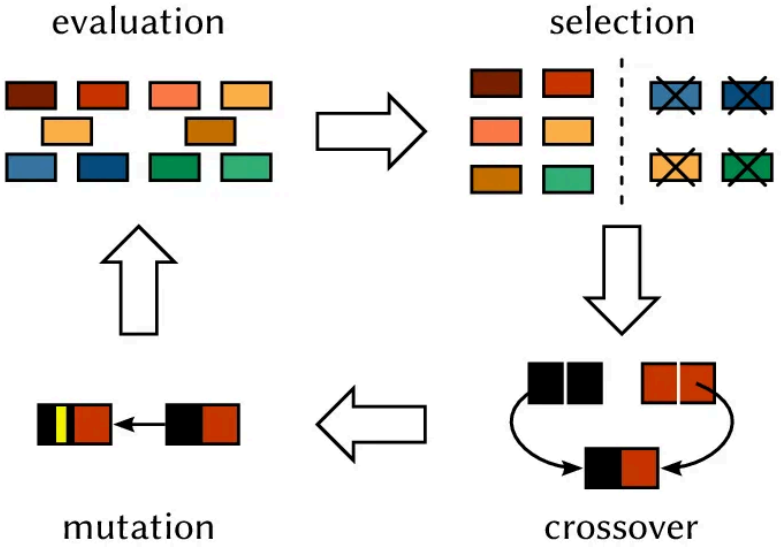

Figura 1: *Transformações em um AG*.

1.2. MÉTODOS

1.2.1. Representação dos Dados

A primeira etapa desenvolvida foi a representação dos dados, foi realizada uma busca no portal do aluno na matriz acadêmica do curso de Ciência da Computação da UTFPR - Campus Santa Helena de modo a obter os dados necessários para a representação e categorização do problema dessa forma foi desenvolvido uma planilha contendo uma tabela com as disciplinas, disponível também neste link Representacao do Dados

Periodo	codigo	Disciplina	Dias Turno Hora e Sala
			Exemplo 6T3E09 (6 = sexta, T = tarde, 3 = Terceiro Horario, E09= sala)
1	CC1ICC	Introdução À Ciência Da Computação	6T3(E09) - 6T4(E09) - 6T5(E09)
1	MA3AL	Álgebra Linear	3T3(L10) - 3T4(L10) - 3T5(L10)
1	CC1MBD	Modelagem De Banco De Dados	4T4(L5) - 4T5(L5) - 5T4(L5) - 5T5(L5)
1	HU1LA	Linguagem Acadêmica	2M2(K7) - 2M3(K7)
1	MA1FM	Fundamentos De Matemática	3T1(L10) - 3T2(L10) - 6T1(L10) - 6T2(L10)
1	MA1LM	Lógica Matemática	3T3(L8) - 3T4(L8) - 3T5(L8)
1	CC1AED1	Algoritmos E Estrutura De Dados I	4T1(L5) - 4T2(L5) - 4T3(L5) - 5T1(L5) - 5T2(L5) - 5T3(L5)
2	CC2CLD	Circuitos Lógicos Digitais	3M2(E09) - 3M2(E09) - 3M3(E09) - 3M3(E09) - 5M4(E09) - 5M4(E09) - 5M5(E09) - 5M5(E09)
2	CC2ER	Engenharia De Requisitos	3M1(L5) - 6M3(L5) - 6M4(L5) - 6M5(L5)
2	CC2POO	Programação Orientada A Objetos	2M2(E09) - 2M3(E09) - 4M1(L5) - 4M2(L5)
2	MA2MA	Matemática Aplicada	3M4(L8) - 3M5(L8) - 4M4(L8) - 4M5(L8)
2	CC2AED2	Algoritmos E Estrutura De Dados II	4M3(L5) - 5M1(L5) - 5M2(L5) - 5M3(L5)
3	CC3AED3	Algoritmos E Estrutura De Dados Iii	2T1(L5) - 2T2(L5) - 3T1(L5) - 3T2(L5)
3	CC3AOC	Arquitetura E Organização De Computadores	4T1(E09) - 4T2(E09) - 4T3(E09) - 5T1(E09) - 5T2(E09) - 5T3(E09)
3	CC3ES	Engenharia De Software	4T4(E10) - 4T5(E10) - 5T4(E10) - 5T5(E10)
3	MA3AL	Álgebra Linear	3T3(L10) - 3T4(L10) - 3T5(L10)
3	MA3CA	Cálculo Aplicado	2T4(L8) - 2T5(L8) - 6T1(L8) - 6T2(L8)
3	CC3PI1	Projeto Integrador I: Computação E Educação	'T6(L10) - 3T6(L10) - 4T6(L10) - 5T6(L10) - 6T5(L10) - 6T6(L10) - 7M1(L10) - 7M2(L10) - 7M3(L10) - 7M4(L10) - 7M5(L10) - 7M6(L10)
4	CC4IHC	Interação Humano Computador	3M2(L4) - 3M2(L4) - 3M3(L4) - 3M3(L4) - 6M1(L4) - 6M1(L4) - 6M2(L4) - 6M2(L4)
4	CC4PO	Pesquisa E Ordenação	3M4(L4) - 3M5(L4) - 4M4(L4) - 4M5(L4)
4	CC4LBD	Laboratório De Banco De Dados	4M1(L6) - 4M2(L6) - 5M4(L6) - 5M5(L6)
4	CC4RC	Redes De Computadores	4M3(E10) - 5M1(E10) - 5M2(E10) - 5M3(E10)
4	CC4SO	Sistemas Operacionais	5T4(L2B) - 5T5(L2B) - 6T4(L2B) - 6T5(L2B)
4	MA4PE	Probabilidade E Estatística	3M1(L6) - 6M3(L6) - 6M4(L6) - 6M5(L6)
4	CC24A	Linguagem De Programação Objeto Orientada	2T2(L6) - 2T3(L6) - 4T2(L6) - 4T3(L6)

Professores		Carga Horaria Semanal
Thiago	Prof.1	40/hrs
Evandro	Prof.2	40/hrs
Davi	Prof.3	40/hrs
Gilvani	Prof.4	40/hrs
Giani	Prof.5	40/hrs
Gloria	Prof.6	40/hrs
Victor	Prof.7	40/hrs
Claudio	Prof.8	40/hrs
Carlos	Prof.9	40/hrs
Euclides	Prof.10	40/hrs
Leiliane	Prof.11	40/hrs

Salas
L10
L08
L09
E09
L5
E10
L4

Figura 2: *Representação dos Dados* .

1.2.2. Funcionamento do Algoritmo Genético

Como já descrito anteriormente o os algoritmos genéticos operam por meio de um processo interativo que simula a evolução das populações, deste modo estruturamos o código de acordo com as etapas descritas anteriormente.

1. **Representação dos Dados**:
    - As disciplinas, salas e professores são representados por dicionários e listas, contendo informações sobre códigos de disciplinas, dias, turnos, horas, salas disponíveis e professores.
    - As disciplinas são descritas com suas restrições específicas de dias e horários.

```
# Dados das disciplinas e salas
disciplinas = [
    {"codigo": "CC1ICC", "dias": [6, 6, 6], "turno": "T", "horas": [3, 4, 5]},
    {"codigo": "MA3AL", "dias": [3, 3, 3], "turno": "T", "horas": [3, 4, 5]},
    {"codigo": "CC1MBD", "dias": [4, 4, 5, 5], "turno": "T", "horas": [4, 5, 4, 5]},
    {"codigo": "HU1LA", "dias": [2, 2], "turno": "M", "horas": [2, 3]},
    {"codigo": "MA1FM", "dias": [3, 3, 6, 6], "turno": "T", "horas": [1, 2, 1, 2]},
    {"codigo": "MA1LM", "dias": [3, 3, 3], "turno": "T", "horas": [3, 4, 5]},
    {"codigo": "CC1AED1", "dias": [4, 4, 4, 5, 5, 5], "turno": "T", "horas": [1, 2, 3, 1, 2, 3]},
    {"codigo": "CC2CLD", "dias": [3, 3, 3, 3, 5, 5, 5, 5], "turno": "M", "horas": [2, 2, 3, 3, 4, 4, 5, 5]},
    {"codigo": "CC2ER", "dias": [3, 6, 6, 6], "turno": "M", "horas": [1, 3, 4, 5]},
    {"codigo": "CC2POO", "dias": [2, 2, 4, 4], "turno": "M", "horas": [2, 3, 1, 2]},
    {"codigo": "MA2MA", "dias": [3, 3, 4, 4], "turno": "M", "horas": [4, 5, 4, 5]},
    {"codigo": "CC2AED2", "dias": [4, 5, 5, 5], "turno": "M", "horas": [3, 1, 2, 3]},
    {"codigo": "CC3AED3", "dias": [2, 2, 3, 3], "turno": "T", "horas": [1, 2, 1, 2]},
    {"codigo": "CC3AOC", "dias": [4, 4, 4, 5, 5, 5], "turno": "T", "horas": [1, 2, 3, 1, 2, 3]},
    {"codigo": "CC3ES", "dias": [4, 4, 5, 5], "turno": "T", "horas": [4, 5, 4, 5]},
    {"codigo": "MA3AL", "dias": [3, 3, 3], "turno": "T", "horas": [3, 4, 5]},
    {"codigo": "MA3CA", "dias": [2, 2, 6, 6], "turno": "T", "horas": [4, 5, 1, 2]},
    {"codigo": "CC3PI1", "dias": [2, 3, 4, 5, 6, 6, 7, 7, 7, 7, 7, 7], "turno": "T", "horas": [6, 6, 6, 6, 5, 6, 1, 2, 3, 4, 5, 6]},
    {"codigo": "CC4IHC", "dias": [3, 3, 3, 3, 6, 6, 6, 6], "turno": "M", "horas": [2, 2, 3, 3, 1, 1, 2, 2]},
    {"codigo": "CC4PO", "dias": [3, 3, 4, 4], "turno": "M", "horas": [4, 5, 4, 5]},
    {"codigo": "CC4LBD", "dias": [4, 4, 5, 5], "turno": "M", "horas": [1, 2, 4, 5]},
    {"codigo": "CC4RC", "dias": [4, 5, 5, 5], "turno": "M", "horas": [3, 1, 2, 3]},
    {"codigo": "CC4SO", "dias": [5, 5, 6, 6], "turno": "T", "horas": [4, 5, 4, 5]},
    {"codigo": "MA4PE", "dias": [3, 6, 6, 6], "turno": "M", "horas": [1, 3, 4, 5]},
    {"codigo": "CC24A", "dias": [2, 2, 4, 4], "turno": "T", "horas": [2, 3, 2, 3]},
]

salas = ["L10", "L08", "L09", "E09", "L5", "E10", "L4"]
professores = [f"prof{i}" for i in range(1, 12)]
```

2. **Criação de Indivíduos**:
    - Um indivíduo representa uma grade de horários completa, onde cada disciplina é atribuída a uma sala e um professor aleatório.
    - A função create_individual() gera esses indivíduos, garantindo que cada um contenha todas as disciplinas com alocações iniciais aleatórias.

```
# Representação de um indivíduo como um dicionário de horários
def create_individual():
    return {d["codigo"]: {"codigo": d["codigo"], "dias": d["dias"], "turno": d["turno"], "horas": d["horas"], "sala": random.choice(salas), "professor":
random.choice(professores)} for d in disciplinas}
```

3. **Função de Aptidão**:
    - A função fitness(individual) avalia a qualidade de cada indivíduo, penalizando conflitos de horários entre disciplinas na mesma sala e excesso de horas atribuídas a professores.
    - A função também contabiliza o total de horas que cada professor está alocado, penalizando alocações que excedem 40 horas semanais.

```python
# Função de aptidão: penaliza conflitos de horários e salas, e excesso de horas dos professores
def fitness(individual):
    score = 0
    horario_ocupado = {}
    horas_professores = {prof: 0 for prof in professores}

    for disc in individual.values():
        for dia, hora in zip(disc["dias"], disc["horas"]):
            chave = (dia, disc["turno"], hora, disc["sala"])
            if chave in horario_ocupado:
                score -= 1  # Penaliza conflitos de sala
            else:
                horario_ocupado[chave] = disc["codigo"]
            horas_professores[disc["professor"]] += 1
            if horas_professores[disc["professor"]] > 40:
                score -= (horas_professores[disc["professor"]] - 40)  # Penaliza excesso de horas do professor
    return score
```

4. **Seleção por Torneio**:
    - O método de seleção por torneio é utilizado para escolher os indivíduos que irão cruzar e gerar a próxima geração.
    - A função tournament_selection(population) seleciona os melhores indivíduos entre grupos aleatórios da população atual.

```python
# Seleção por torneio
def tournament_selection(population):
    best = random.choice(population)
    for _ in range(2):  # Torneio de tamanho 3
        ind = random.choice(population)
        if fitness(ind) > fitness(best):
            best = ind
    return best
```

5. **Crossover**:
    - A função crossover(parent1, parent2) combina dois indivíduos (pais) para criar um novo indivíduo (filho), misturando as alocações de horários das disciplinas de ambos os pais.

```python
# Crossover: troca de disciplinas entre dois indivíduos
def crossover(parent1, parent2):
    child = {}
    for key in parent1.keys():
        if random.random() > 0.5:
            child[key] = parent1[key]
        else:
            child[key] = parent2[key]
```

```
    return child
```

6. **Mutação**:
    - A função mutate(individual) aplica mutações aleatórias nos indivíduos, alterando os horários, salas ou professores de algumas disciplinas para introduzir variabilidade na população.

```
# Mutação: modifica aleatoriamente um horário, sala ou professor de uma disciplina
def mutate(individual):
    for disc in individual.values():
        if random.random() < MUTATION_RATE:
            i = random.randint(0, len(disc["dias"]) - 1)
            disc["horas"][i] = random.randint(1, 6)  # Hora aleatória
            disc["dias"][i] = random.randint(2, 6)  # Dia aleatório
            disc["sala"] = random.choice(salas)  # Sala aleatória
            disc["professor"] = random.choice(professores)  # Professor aleatório
```

7. **Algoritmo Genético**:
    - A função genetic_algorithm() gerencia o processo evolutivo, inicializando a população, aplicando seleção, crossover e mutação ao longo de várias gerações, e retornando o melhor indivíduo encontrado.

```
# Algoritmo Genético
def genetic_algorithm():
    population = [create_individual() for _ in range(POPULATION_SIZE)]
    for generation in range(GENERATIONS):
        new_population = []
        for _ in range(POPULATION_SIZE):
            parent1 = tournament_selection(population)
            parent2 = tournament_selection(population)
            child = crossover(parent1, parent2)
            mutate(child)
            new_population.append(child)
        population = new_population
        best_individual = max(population, key=fitness)
        print(f"Generation {generation}: Best Fitness = {fitness(best_individual)}")
    return best_individual
```

1.2.3. Medidas de Interação

Para estabelecer uma métrica para as gerações definimos número de interações como 1000.

```
# Parâmetros do algoritmo genético
POPULATION_SIZE = 100
GENERATIONS = 1000
MUTATION_RATE = 0.01
```

Link Colab para o Código Algoritmo Genetico - Grade de Horarios.ipynb

2.2.3.6 Testes

O primeiro teste feito foi usado as configurações mínimas propostas no trabalho com 500 iterações

```
Generation 499: Best Fitness = -25
Melhor grade de horários encontrada:
CC1ICC: Dias [2, 3, 4] Turno T Horas [5, 2, 6] Sala L09 Professor prof1
MA3AL: Dias [4, 3, 6] Turno T Horas [5, 3, 5] Sala L09 Professor prof6
CC1MBD: Dias [5, 5, 5, 6] Turno T Horas [5, 2, 1, 6] Sala E10 Professor prof5
HU1LA: Dias [2, 2] Turno M Horas [3, 3] Sala L10 Professor prof9
MA1FM: Dias [3, 5, 6, 5] Turno T Horas [4, 4, 3, 4] Sala L09 Professor prof6
MA1LM: Dias [2, 6, 6] Turno T Horas [6, 1, 1] Sala E09 Professor prof4
CC1AED1: Dias [5, 6, 3, 3, 6, 2] Turno T Horas [4, 5, 2, 5, 1, 2] Sala L09 Professor prof10
CC2CLD: Dias [6, 6, 5, 6, 6, 5, 5, 3] Turno M Horas [5, 2, 5, 4, 1, 1, 5, 5] Sala E09 Professor prof4
CC2ER: Dias [3, 4, 6, 3] Turno M Horas [6, 2, 4, 1] Sala L09 Professor prof5
CC2POO: Dias [6, 2, 6, 6] Turno M Horas [1, 6, 5, 1] Sala E09 Professor prof8
MA2MA: Dias [6, 2, 6, 6] Turno M Horas [4, 1, 2, 3] Sala E09 Professor prof8
CC2AED2: Dias [3, 2, 5, 2] Turno M Horas [2, 3, 2, 5] Sala L10 Professor prof1
CC3AED3: Dias [2, 3, 3, 2] Turno T Horas [6, 6, 1, 5] Sala L08 Professor prof5
CC3AOC: Dias [2, 4, 2, 3, 2, 5] Turno T Horas [2, 5, 4, 3, 3, 4] Sala L4 Professor prof1
CC3ES: Dias [3, 5, 5, 3] Turno T Horas [6, 3, 3, 2] Sala L4 Professor prof9
MA3CA: Dias [6, 4, 4, 5] Turno T Horas [4, 1, 2, 1] Sala L09 Professor prof6
CC3PI1: Dias [3, 3, 6, 5, 5, 3, 5, 6, 5, 3, 5, 4] Turno T Horas [6, 2, 3, 5, 4, 4, 4, 1, 2, 1, 5, 5] Sala L4 Professor prof3
CC4IHC: Dias [4, 4, 2, 4, 3, 6, 2, 5] Turno M Horas [4, 2, 6, 1, 5, 1, 2, 1] Sala L5 Professor prof3
CC4PO: Dias [6, 2, 6, 4] Turno M Horas [1, 5, 5, 1] Sala L5 Professor prof3
CC4LBD: Dias [4, 2, 2, 6] Turno M Horas [2, 3, 6, 2] Sala L10 Professor prof4
CC4RC: Dias [4, 2, 5, 4] Turno M Horas [5, 2, 5, 3] Sala L5 Professor prof8
CC4SO: Dias [6, 2, 2, 5] Turno T Horas [6, 2, 6, 3] Sala L08 Professor prof7
MA4PE: Dias [5, 2, 3, 4] Turno M Horas [6, 5, 3, 6] Sala E10 Professor prof9
CC24A: Dias [6, 3, 5, 4] Turno T Horas [4, 6, 6, 5] Sala E10 Professor prof1
```

Figura 9: Resultado do teste com 500 iterações.

Com o segundo teste foi usado o dobro de interações totalizando 1000 interações

```
Melhor grade de horários encontrada:
CC1ICC: Dias [2, 2, 5] Turno T Horas [5, 4, 3] Sala E09 Professor prof1
MA3AL: Dias [5, 6, 6] Turno T Horas [5, 5, 4] Sala L08 Professor prof1
CC1MBD: Dias [2, 4, 2, 2] Turno T Horas [3, 4, 4, 4] Sala L09 Professor prof6
HU1LA: Dias [5, 4] Turno M Horas [2, 6] Sala L10 Professor prof7
MA1FM: Dias [6, 5, 5, 2] Turno T Horas [6, 1, 4, 5] Sala E10 Professor prof11
MA1LM: Dias [3, 6, 3] Turno T Horas [3, 3, 3] Sala L08 Professor prof8
CC1AED1: Dias [4, 3, 4, 6, 4, 6] Turno T Horas [1, 3, 2, 6, 3, 3] Sala E10 Professor prof5
CC2CLD: Dias [6, 3, 6, 6, 5, 6, 5, 3] Turno M Horas [5, 6, 3, 4, 2, 5, 2, 1] Sala L10 Professor prof10
CC2ER: Dias [3, 3, 4, 6] Turno M Horas [2, 1, 4, 2] Sala L09 Professor prof9
CC2POO: Dias [6, 3, 3, 4] Turno M Horas [6, 1, 3, 1] Sala L4 Professor prof3
MA2MA: Dias [5, 2, 4, 3] Turno M Horas [1, 6, 5, 2] Sala L5 Professor prof3
CC2AED2: Dias [3, 3, 3, 3] Turno M Horas [1, 2, 1, 3] Sala L10 Professor prof3
CC3AED3: Dias [5, 6, 6, 6] Turno T Horas [3, 1, 6, 4] Sala L09 Professor prof7
CC3AOC: Dias [3, 2, 6, 6, 2, 2] Turno T Horas [3, 5, 4, 2, 4, 1] Sala L08 Professor prof10
CC3ES: Dias [3, 3, 4, 2] Turno T Horas [6, 6, 2, 1] Sala L10 Professor prof8
MA3CA: Dias [6, 5, 5, 4] Turno T Horas [4, 1, 1, 5] Sala L4 Professor prof1
CC3PI1: Dias [3, 6, 4, 3, 6, 5, 3, 3, 3, 3, 6, 5] Turno T Horas [6, 3, 3, 1, 3, 4, 2, 1, 5, 4, 1, 2] Sala L5 Professor prof7
CC4IHC: Dias [4, 3, 2, 2, 5, 2, 6, 5] Turno M Horas [1, 1, 5, 3, 6, 6, 5, 4] Sala L09 Professor prof8
CC4PO: Dias [5, 4, 2, 6] Turno M Horas [1, 4, 5, 3] Sala E09 Professor prof3
CC4LBD: Dias [4, 4, 4, 2] Turno M Horas [1, 2, 3, 1] Sala L10 Professor prof7
CC4RC: Dias [3, 6, 2, 2] Turno M Horas [2, 2, 1, 5] Sala E10 Professor prof5
CC4SO: Dias [6, 5, 2, 6] Turno T Horas [4, 6, 2, 6] Sala L10 Professor prof1
MA4PE: Dias [6, 3, 4, 5] Turno M Horas [3, 1, 5, 5] Sala L08 Professor prof11
CC24A: Dias [3, 3, 2, 5] Turno T Horas [5, 4, 6, 2] Sala L4 Professor prof5
```

Figura 3: Resultado do teste com 1000 interações.

Para o segundo teste notei que os dados mais distribuídos o que pode ser bom levando em consideração o problema proposto no entanto os horários encontram-se quebrados acredito que uma análise minuciosa na forma de representação dos dias e das

horas de aula possa solucionar esse problema de modo a distribuir as aulas de forma conjunta nos seus respectivos dias.

```
Melhor grade de horários encontrada:
CC1ICC: Dias [2, 2, 5] Turno T Horas [5, 4, 3] Sala E09 Professor prof1
MA3AL: Dias [5, 6, 6] Turno T Horas [5, 5, 4] Sala L08 Professor prof1
CC1MBD: Dias [2, 4, 2, 2] Turno T Horas [3, 4, 4, 4] Sala L09 Professor prof6
HU1LA: Dias [5, 4] Turno M Horas [2, 6] Sala L10 Professor prof7
MA1FM: Dias [6, 5, 5, 2] Turno T Horas [6, 1, 4, 5] Sala E10 Professor prof11
MA1LM: Dias [3, 6, 3] Turno T Horas [3, 3, 3] Sala L08 Professor prof8
CC1AED1: Dias [4, 3, 4, 6, 4, 6] Turno T Horas [1, 3, 2, 6, 3, 3] Sala E10 Professor prof5
CC2CLD: Dias [6, 3, 6, 6, 5, 6, 5, 3] Turno M Horas [5, 6, 3, 4, 2, 5, 2, 1] Sala L10 Professor prof10
CC2ER: Dias [3, 3, 4, 6] Turno M Horas [2, 1, 4, 2] Sala L09 Professor prof9
CC2POO: Dias [6, 3, 3, 4] Turno M Horas [6, 1, 3, 1] Sala L4 Professor prof3
MA2MA: Dias [5, 2, 4, 3] Turno M Horas [1, 6, 5, 2] Sala L5 Professor prof3
CC2AED2: Dias [3, 3, 3, 3] Turno M Horas [1, 2, 1, 3] Sala L10 Professor prof3
CC3AED3: Dias [5, 6, 6, 6] Turno T Horas [3, 1, 6, 4] Sala L09 Professor prof7
CC3AOC: Dias [3, 2, 6, 6, 2, 2] Turno T Horas [3, 5, 4, 2, 4, 1] Sala L08 Professor prof10
CC3ES: Dias [3, 3, 4, 2] Turno T Horas [6, 6, 2, 1] Sala L10 Professor prof8
MA3CA: Dias [6, 5, 5, 4] Turno T Horas [4, 1, 1, 5] Sala L4 Professor prof1
CC3PI1: Dias [3, 6, 4, 3, 6, 5, 3, 3, 3, 3, 6, 5] Turno T Horas [6, 3, 3, 1, 3, 4, 2, 1, 5, 4, 1, 2] Sala L5 Professor prof7
CC4IHC: Dias [4, 3, 2, 2, 5, 2, 6, 5] Turno M Horas [1, 1, 5, 3, 6, 6, 5, 4] Sala L09 Professor prof8
CC4PO: Dias [5, 4, 2, 6] Turno M Horas [1, 4, 5, 3] Sala E09 Professor prof3
CC4LBD: Dias [4, 4, 4, 2] Turno M Horas [1, 2, 3, 1] Sala L10 Professor prof7
CC4RC: Dias [3, 6, 2, 2] Turno M Horas [2, 2, 1, 5] Sala E10 Professor prof5
CC4SO: Dias [6, 5, 2, 6] Turno T Horas [4, 6, 2, 6] Sala L10 Professor prof1
MA4PE: Dias [6, 3, 4, 5] Turno M Horas [3, 1, 5, 5] Sala L08 Professor prof11
CC24A: Dias [3, 3, 2, 5] Turno T Horas [5, 4, 6, 2] Sala L4 Professor prof5
```

Figura 4: Resultado do teste com 2000 interações.

2. RESULTADOS E DISCUSSÃO

Com a análise dos resultados foi possível ver que de acordo com o número de interações e seu peso gerado pela função fitness os dados têm um agrupamento melhor em comparação com os valores abaixo de 1000 interações durante algumas análises aloquei as interações para 1281 onde notei que o pesos ficavam próximos a -1 , foi possível notar o desempenho bom do AG em relação a esse problema de combinação.

3.1 REFERÊNCIAS

Artigo em Website:

JOHN HENRY HOLLAND. *Encyclopaedia Britannica*. Disponível em: https://www.britannica.com/biography/John-Henry-Holland. Acesso em: 3 jun. 2024.

PASSING OF PROF. JOHN HOLLAND, FATHER OF GENETIC ALGORITHMS AND PIONEER IN COMPLEX SYSTEMS. *BEACON Center for the Study of Evolution in Action*. Disponível em: https://www.beacon-center.org/blog/2015/08/09/passing-of-prof-john-holland-father-of-genetic-algorithms-and-pioneer-in-complex-systems. Acesso em: 3 jun. 2024.

COMPUTER SCIENCE PIONEER DIES. *The Scientist Magazine*. Disponível em: https://www.the-scientist.com/news-opinion/computer-science-pioneer-dies-34635. Acesso em: 3 jun. 2024.

https://medium.com/cecosmos/programa%C3%A7%C3%A3o-gen%C3%A9tica-a-sele%C3%A7%C3%A3o-natural-de-f%C3%B3rmula-edc316dfb0fe

https://repositorio.ufpb.br/jspui/bitstream/123456789/3824/1/JLLS01122016.pdf