Algoritmo Genético - Alocação de horários

BCC | Inteligência Artificial 2022.2
Projeto – Algoritmos genéticos – Grade de horário
Luiz Davi e Thiago Cavalcanti

UFAPE

Mostre no seu código todos os seguintes elementos:

Representação (definição do indivíduo):

O indivíduo é uma grade de horários, representado por uma lista de matrizes, onde cada matriz representa um período do curso.

No nosso caso, usamos o curso de BCC como referência. Logo, são 9 períodos e dois horários noturnos (18:30h – 20:10h, 20:10h – 21:50h).

A matriz tem tamanho 2x5, onde as duas linhas representam os dois horários, enquanto as cinco colunas são os dias letivos (segunda a sexta). Entretanto, são parâmetros flexíveis, podendo ser variáveis.

Função de avaliação (função de aptidão)

É representada pela função calcular_aptidao, onde analisa a grade para verificar se as restrições estão sendo respeitadas.

Em caso de choque de horário, uma enorme pontuação é retirada da aptidão, dessa forma, a grade será descartada da população.

Se outra restrição for descumprida, como o professor lecionar em um dia que ele não escolheu dar aulas, terá um peso menor, pois a grade ainda é viável.

```
def calcular_aptidao(grade):
    aptidao = MELHOR_APTIDAO

    # Criar um dicionário para armazenar os horários de cada docente
    horarios_docentes = { docente.nome: set() for docente in DOCENTES }

    # Percorrer a grade de horários verificando se há choques
    for periodo in grade:
        for dia in range(QUANT_DE_DIAS):
            for horario in range(QUANT_DE_HORARIOS):
                disciplina = periodo[horario][dia]
                if disciplina:
                    for docente in disciplina.docentes:
                        # validaçao choque de horário
                        if (dia, horario) in horarios_docentes[docente.nome]:
                            # Choque de horário, remover pontuação
                            aptidao -= PONTUACAO_CHOQUE
                            #print(f"Choque no dia {LISTA_DIAS[dia]} às {LISTA_HORARIOS[horario]} do professor {docente.nome}")
                            break
                        # validaçao dos dias não lecionado preferidos pelo professor
                        else:
                            horarios_docentes[docente.nome].add((dia, horario))
                            # validacao dos horarios dos professores
                            dia_bom = True
                            for dia_nao_lecionavel in docente.dias_sem_lecionar:
                                if dia_nao_lecionavel == dia:
                                  aptidao -= PONTUACAO_DIA_NAO_LECIONAVEL
                                  dia_bom = False
                                  #print(f"Docente {docente.nome} lecionando no dia {LISTA_DIAS[dia]} às {LISTA_HORARIOS[horario]}")
                                  break
                            if dia_bom:
```

```
                                aptidao += 2

    # validacao de dias consecutivos
    for docente in DOCENTES:
        if docente.aulas_concentradas:
          concentracao = []
          horarios = horarios_docentes[docente.nome]

          for horario in horarios:
            concentracao.append(horario[0])

          if not verificar_concentracao(sorted(concentracao)):
            aptidao -= PONTUACAO_AULAS_CONCENTRADAS

    return aptidao
```

População

Uma lista que contém todos os indivíduos (grades de horário), com tamanho máximo de 60 indivíduos. Porém, também é um parâmetro variável para o algoritmo.

Mecanismo de seleção dos pais

Seleciona pares de grades consecutivos dois a dois, ou seja, seleciona pais vizinhos.

```
while melhor['geracao'] < QUANT_GERACOES_SEM_MELHORIA:
  novas_grades = []
  for i in range(0, len(populacao), 2):
      grade1 = populacao[i]
      grade2 = populacao[i+1]

      novas_grades.append(crossover(grade1, grade2))
```

Operadores de variação

Recombinação (crossover)

O cruzamento entre os pais é feito copiando cada período aleatoriamente de um dos dois. Os filhos são gerados combinando um período inteiro para não correr o risco de perder alguma aula durante o corte. Para cada período, é sorteado um dos dois pais, assim, mantém-se a diversidade genética.

```
def crossover(grade1, grade2):
  grade_filha = []

  for index in range(QUANT_DE_PERIODOS):
    grade_pai = random.choice([grade1, grade2])
    grade_filha.append(grade_pai[index])

  for _ in range(QUANT_MUTACOES):
    mutacao(grade_filha)

  return grade_filha
```

Mutação

A cada cruzamento há uma quantidade de mutações, que também pode ser variável.

A cada mutação, um período é sorteado e é montado novamente.

Dessa forma, todas as grades estão sofrendo mutações.

```
def mutacao(grade):
  periodo_mutado = random.randint(1, QUANT_DE_PERIODOS)

  novo_periodo = [[None for _ in range(QUANT_DE_DIAS)] for _ in range(QUANT_DE_HORARIOS)]

  preencher_horario(periodo_mutado, novo_periodo, DISCIPLINAS)

  grade[periodo_mutado - 1] = novo_periodo
```

Mecanismo de seleção dos sobreviventes

A cada geração, os X indivíduos menos aptos da população são trocados pelos filhos caso as aptidões deles sejam maiores.

```
def thanos(novas_grades, populacao):
  for nova_grade in novas_grades:
      grade_ruim = menos_apto(populacao)
      nova_aptidao = calcular_aptidao(nova_grade)
      if nova_aptidao > grade_ruim['aptidao']:
          populacao[grade_ruim['posicao']] = nova_grade
```

Inicialização da população

Os indivíduos iniciais são gerados aleatoriamente, através da função criar_populacao_inicial(disciplinas), formando a população. Estes serão os primeiros pais, que passarão a ser combinados.

Para cada período, as disciplinas serão combinadas para formar seus horários. Por fim, teremos a grade de um curso. Isso se repete até que o tamanho máximo da população seja atingido.

```
def preencher_horario(periodo, matriz_horario, disciplinas):
    horarios_disponiveis = QUANT_HORARIOS_DISPO  # Cada período tem 10 horários disponíveis
    disciplinas_do_periodo = [disc for disc in disciplinas if disc.periodo == periodo]
    random.shuffle(disciplinas_do_periodo)  # Embaralha as disciplinas do período

    for disciplina in disciplinas_do_periodo:
        aulas_restantes = disciplina.quantidade_de_aulas
        for horario in range(QUANT_DE_HORARIOS):
            for dia in range(QUANT_DE_DIAS):
                if aulas_restantes > 0 and matriz_horario[horario][dia] == None:
                    matriz_horario[horario][dia] = disciplina
                    aulas_restantes -= 1
                    horarios_disponiveis -= 1
                    if aulas_restantes == 0:
                        break
            if aulas_restantes == 0:
                break
        if horarios_disponiveis == 0:
            break


def criar_populacao_inicial(disciplinas):
    populacao = []

    for _ in range(QUANT_DA_POPULACAO_INICIAL):
        periodos = []

        # Preenche as matrizes de horários para cada período
        for periodo in range(1, QUANT_DE_PERIODOS + 1):
            # inicializa o período com valores nulos
            matriz_horario = [[None for _ in range(QUANT_DE_DIAS)] for _ in range(QUANT_DE_HORARIOS)]

            preencher_horario(periodo, matriz_horario, disciplinas)

            # Atualiza diretamente a matriz de horário do período
            periodos.append(matriz_horario)

        populacao.append(periodos)

    return populacao
```

Término (condição de parada)

Periodo 1	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	logica matematica	logica matematica	geometria analitica	geometria analitica	introducao a computacao
20:10h - 21:50h	calculo 1	calculo 1	introducao a programacao	introducao a programacao	introducao a programacao

Periodo 2	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	calculo 2	calculo 2	algebra linear	algebra linear	aed 1
20:10h - 21:50h	aed 1	poo	poo	fisica	fisica

Periodo 3	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	probabilidade e estatistica	probabilidade e estatistica	metodologia cientifica	metodologia cientifica	aed 2
20:10h - 21:50h	aed 2	sistemas digitais	sistemas digitais	matematica discreta	matematica discreta

Periodo 4	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	banco de dados	banco de dados	plp	plp	arquitetura de computadores
20:10h - 21:50h	arquitetura de computadores	engenharia de software	engenharia de software	paa	paa

Periodo 5	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	sistemas de informacao	sistemas de informacao	sistemas operacionais	sistemas operacionais	redes de computadores
20:10h - 21:50h	redes de computadores	inteligencia artificial	inteligencia artificial	teoria da computacao	teoria da computacao

Periodo 6	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	compiladores	compiladores	computacao grafica	computacao grafica	sistemas distribuidos

Periodo 7	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	projetao	projetao	projetao	comp e soc	ihc
20:10h - 21:50h	ihc	optativa 1	optativa 1	optativa 2	optativa 2

Periodo 8	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	optativa 7	optativa 7	optativa 3	optativa 3	optativa 5
20:10h - 21:50h	optativa 5	optativa 6	optativa 6	optativa 4	optativa 4

Periodo 9	Segunda	Terca	Quarta	Quinta	Sexta
18:30h - 20:10h	-	-	tcc	eso	-
20:10h - 21:50h	-	-	-	-	-

Total de gerações: 417
Tempo de execução: 0 minuto(s) e 49 segundo(s)

Resultado de uma execução