Entrega Final de Computação Evolutiva

Aluno: Roxanne Silva Julia - 11711BCC022

Aluno: Lucas Yudi Matsuhashi - 11711BCC031

Aluno: Pedro Henrique Bufulin de Almeida 11711BCC028

Aluno: Ricardo Zamboni Silva - 11821BCC004

Título do Trabalho: Algoritmo Genético para o problema dos Filmes vencedores do Oscar

1 Representação do cromossomo

Cada cromossomo foi representado com uma lista de inteiros em que cada valor inteiro representa um filme.

Exemplo: 1 - Cimarron, 2 - Dances with wolves

A ordem dos inteiros no vetor corresponde à ordem de visualização dos filmes. Ou seja, o inteiro na posição 0 do vetor representa o primeiro filme a ser visto; o inteiro na posição 1 do vetor representa o segundo filme a ser visto e assim por diante.

O comprimento do cromossomo corresponde à quantidade total de filmes e cada gene representa um filme e quando ele será visto.

Exemplo de cromossomo: [71, 58, 31, 35, 80, 7, 67, 16, 8, 89, 66, 79, 33, 64, 91, 36, 69, 53, 62, 51, 82, 50, 2, 14, 73, 4, 17, 12, 92, 32, 83, 76, 6, 11, 61, 75, 26, 37, 68, 1, 39, 20, 81, 54, 44, 63, 93, 88, 42, 52, 40, 29, 74, 57, 47, 19, 9, 10, 65, 60, 77, 70, 90, 55, 30, 59, 13, 25, 15, 85, 86, 78, 72, 46, 41, 38, 49, 18, 45, 28, 22, 21, 5, 84, 87, 27, 23, 48, 34, 43, 3, 24, 56]

2 Função Objetivo

A Função Objetivo consiste em minimizar a quantidade de dias necessários para assistir a todos os filmes e maximizar a diversidade diária de gêneros.

3 Decodificação do cromossomo

A função de decodificação do cromossomo é apresentada, com comentários, nas Figuras 1 e 2:

4 Avaliação do indivíduo

Um indíviduo é avaliado através das funções WorstCromossome e BestCromossome, as quais foram apresentadas, com comentários, na Figura 3 . Um indivíduo é avaliado como o pior da população quando ele, primeiramente, possui o maior número de dias e, segundamente, possui a menor diversidade diária de gêneros. Inversamente, Um indivíduo é avaliado como o melhor da população quando

```python
def fitness (c, time):
    #A funcao fitness recebe um cromossomo e o tempo diário limite (no caso 4 horas) e calcula a quantidade
    # de dias necessários para assistir aos filmes do cromossomo crm e a soma da diversidade diária
    # de gêneros

    #contador de horas
    countHours = 0
    #contador de dias
    countDays = 1
    #listas para controle de diversidade diária de gêneros
    films_per_day = []
    list_genres = []
    #para cada filme do cromossomo
    for idx in c:
        #recupera tempo de durção do filme
        row = filmesDict[idx]
        hours = int(row[1])
        #acrescenta este tempo de duração ao contador de horas
        countHours = countHours + hours
        #se o contador passa do tempo limite
        if countHours > time:
            #contador de horas é igualado ao tempo de duração do filme (filme atual entra no dia seguinte)
            countHours = hours
            #contador de dias é incrementado
            countDays = countDays + 1
```

Figura 1: Função fitness 1

```python
            #retorna a quantidade diferentes de gênero deste dia
            genres = genre(films_per_day)
            #acrescenta esta quantidade à lista de diversidade diária de gêneros
            list_genres.append(genres)
            films_per_day = []
        #cai no else se o contador não passou ainda do tempo limite
        else:
            films_per_day.append(idx)
    #retorna o número de dias do cromossomo e a soma da diversidade diária de gêneros
    return (countDays, sum(list_genres))
```

Figura 2: Função fitness 2

ele, primeiramente, possui o menor número de dias e, segundamente, possui a maior diversidade diária de gêneros.

5 Operadores

Os operadores utilizados no algoritmo foram a mutação e o elitismo.

5.1 Mutação

A mutação de um cromossomo consiste na seleção aleatória de duas posições (filmes) desse cromossomo e na sua comutação.

Exemplo :

Para o cromossomo: [71, 58, 31, 35, 80, 7, 67, 16, 8, 89, 66, 79, 33, 64, 91, 36, 69, 53, 62, 51, 82, 50, 2, 14, 73, 4, 17, 12, 92, 32, 83, 76, 6, 11, 61, 75, 26, 37, 68, 1, 39, 20, 81, 54, 44, 63, 93, 88, 42,

```python
def worstCromossome(fitnessPop):
    # O pior indivíduo é aquele com, primeiramente, um maior número de dias e, segundamente, uma menor
    # diversidade diária de gêneros

    #Ordena decrescentemente a lista de fitness com base no número de dias
    fitnessPop = sorted(fitnessPop, key=lambda tup: tup[0], reverse=True)
    #Recupera o maior número de dias presente na população
    maxfit = fitnessPop[0][0]
    #Recupera todos os indivíduos com o maior número de dias
    fitnessPop = [i for i in fitnessPop if i[0] == maxfit]
    #Recupera, dentre os indivíduos com o maior número de dias, o com a menor diversidade diárira de gêneros
    crm = sorted(fitnessPop, key=lambda tup: tup[1])[0]
    return crm

def bestCromossome(fitnessPop):
    # O melhor indivíduo é aquele com, primeiramente, um menor número de dias e, segundamente, uma maior
    # diversidade diária de gêneros

    #Ordena crescentemente a lista de fitness com base no número de dias
    fitnessPop = sorted(fitnessPop, key=lambda tup: tup[0])
    #Recupera o menor número de dias presente na população
    minfit = fitnessPop[0][0]
    #Recupera todos os indivíduos com o menor número de dias
    fitnessPop = [i for i in fitnessPop if i[0] == minfit]
    #Recupera, dentre os indivíduos com o menor número de dias, o com a maior diversidade diárira de gêneros
    crm = sorted(fitnessPop, key=lambda tup: tup[1], reverse=True)[-1]
    return crm
```

Figura 3: Funções de avaliação do indivíduo

52, 40, 29, 74, 57, 47, 19, 9, 10, 65, 60, 77, 70, 90, 55, 30, 59, 13, 25, 15, 85, 86, 78, 72, 46, 41, 38, 49, 18, 45, 28, 22, 21, 5, 84, 87, 27, 23, 48, 34, 43, 3, 24, 56]

Se as posições 0 e e 4 são escolhidas aleatoriamente, então o cromossomo, depois de mutado, passa a ser: [80, 58, 31, 35, 71, 7, 67, 16, 8, 89, 66, 79, 33, 64, 91, 36, 69, 53, 62, 51, 82, 50, 2, 14, 73, 4, 17, 12, 92, 32, 83, 76, 6, 11, 61, 75, 26, 37, 68, 1, 39, 20, 81, 54, 44, 63, 93, 88, 42, 52, 40, 29, 74, 57, 47, 19, 9, 10, 65, 60, 77, 70, 90, 55, 30, 59, 13, 25, 15, 85, 86, 78, 72, 46, 41, 38, 49, 18, 45, 28, 22, 21, 5, 84, 87, 27, 23, 48, 34, 43, 3, 24, 56]

A função onde essa mutação é realizada é apresentada, com comentários, na Figura 4. Ficará mais claro na Sessão 6 a motivação por trás dos outros aspectos da função.

5.2 Elitismo

O elitismo presente no algoritmo é ilustrado na estrutura principal do algoritmo, na Figura 5. Ele consiste na remoção dos piores indivíduos na mesma proporção que indivíduos novos são acrescentados.

6 Estrutura do AG

O AG é geracional e a sua estrutura principal é apresentada na Figura 5.

```python
#recebe a população pop e uma probabilidade prob
def mutation(pop, prob):
    list_new_cromossome = []
    #para cada cromossomo da população
    for i in pop:
        #uma probabilidade entre 0 e 1 é gerada
        prob_check = uniform(0, 1)
        #se prob menor ou igual à probabilidade gerada acima
        if prob >= prob_check:
            #uma cópia de cromossomo é gerada
            new_cromossome = i.copy()
            #esta cópia é mutada
            change_positions = sample(range(0, len(i)), 2)
            new_cromossome[change_positions[0]], new_cromossome[change_positions[1]] = new_cromossome[change_p
            new_cromossome = restriction(new_cromossome)
            #esta cópia mutada é acrescentada à lista list_new_cromossome
            list_new_cromossome.append(new_cromossome)
    #lista com as cópias mutadas dos cromossomos é retornada
    return list_new_cromossome
```

Figura 4: Funções de mutação

7 Experimentos e Resultados

Foram feitos 100 testes para cada grupo de parâmetros. Os parâmetros analisados foram: número de gerações, probabilidade de mutação e tamanho da população.

- O número de gerações varia entre [50, 100, 200, 500, 1000, 2000, 5000];
- A probabilidade de mutação varia entre [0.0, 0.2, 0.4, 0.6, 0.8, 1.0];
- O tamanho da população varia entre [10, 20, 30, 50, 70, 100].

Os resultados representam:

- Tempo de execucao: Media +- Desvio Padrao dos tempos de execucao (em s);
- Melhores resultados de cada iteração: Um dicionário em que a chave é uma 2-upla do tipo (melhor resultado de tempo, melhor resultado de tipos de gêneros) e o valor é a quantidade de vezes que esse resultado foi obtido;
- Melhor resultado entre todas as iterações: A melhor 2-upla do dicionário acima e a quantidade de vezes que ela foi alcançada.

7.1 Número de gerações (mut=0.4, popSize = 20)

gen=50:

Tempo de execucao: 0.7926841654499185 +- 0.17105314479035 s

Melhores resultados: (71, 43): 15, (71, 42): 12, (71, 44): 10, (70, 44): 7, (71, 40): 6, (70, 46): 6, (70, 43): 5, (70, 42): 4, (70, 45): 4, (70, 41): 4, (71, 41): 4, (71, 39): 4, (71, 45): 3, (72, 40): 2, (69, 46): 2, (69, 44): 2, (69, 45): 2, (69, 47): 2, (72, 41): 2, (70, 48): 1, (70, 47): 1, (72, 43): 1, (69, 49): 1

Melhor resultado: (69, 49): 1

```python
for x in range(generation):
    #lista de novos cromossomos
    MutatedCrm = []
    #lista de fitness da população
    fitnessPop = []
    #a função mutation é chamada para a população com uma certa probabilidade (no caso 0.6)
    MutatedCrm = mutation(pop, 0.6)
    #recupera o número de cromossomos que foram retornados pela função mutation
    nMutated = len(MutatedCrm)
    #acrescenta os cromossomos mutados na população
    pop.extend(MutatedCrm)
    #para cada cromossomo da população extendida
    for crm in pop:
        #o fitness dele é calculado e acrescentado à fitnessPop
        fitnessPop.append(fitness(crm, time))
    #para o mesmo de número de cromossomos mutados que foram acrscentados à população
    for y in range(nMutated):
        #pior cromossomo da população é calculado
        worstCrm = worstCromossome(fitnessPop)
        #índice do pior cromossomo
        worstCrmIndex = fitnessPop.index(worstCrm)
        #pior cromossomo é retirado da população
        pop.pop(worstCrmIndex)
        #fitness do pior cromossomo é retirado da lista de fitness
        fitnessPop.pop(worstCrmIndex)

    #melhor cromossomo é calculado
    bestCrm = bestCromossome(fitnessPop)
```

Figura 5: Estrutura principal do AG

gen=100

Tempo de execução: 1.4524790676899557 +- 0.27685315912927905 s

Melhores resultados: (69, 48): 15, (69, 47): 11, (68, 50): 9, (68, 49): 9, (68, 48): 8, (69, 46): 7, (68, 51): 6, (70, 47): 6, (70, 46): 6, (68, 47): 5, (69, 49): 5, (68, 46): 3, (67, 51): 2, (67, 50): 2, (67, 52): 2, (70, 44): 2, (69, 50): 1, (69, 45): 1

Melhor resultado: (67, 52): 2

gen=200

Tempo de execução: 2.6643018469500293 +- 0.4968334543713723 s Melhores resultados: (67, 53): 19, (67, 52): 13, (67, 54): 12, (68, 52): 11, (68, 51): 8, (67, 51): 6, (66, 54): 5, (67, 50): 5, (68, 49): 4, (68, 50): 4, (66, 52): 2, (69, 50): 2, (66, 56): 2, (66, 55): 2, (70, 48): 1, (67, 48): 1, (65, 56): 1, (66, 51): 1, (67, 49): 1

Melhor resultado: (65, 56): 1

gen=500

Tempo de execução: 6.177227299220013 +- 0.9169621419849353 s Melhores resultados: (66, 56): 55, (67, 54): 16, (65, 58): 12, (66, 55): 11, (66, 54): 3, (67, 53): 1, (65, 57): 1, (65, 56): 1

Melhor resultado: (65, 58): 12

gen=1000

Tempo de execução: 12.060355021749974 +- 1.581207319020692 s

Melhores resultados: (65, 58): 46, (66, 56): 44, (64, 60): 4, (65, 56): 2, (65, 57): 2, (66, 55): 1, (67, 54): 1

Melhor resultado: (64, 60): 4

gen=2000

Tempo de execução: 23.701438712390154 +- 2.6317258899859386 s

Melhores resultados: (65, 58): 79, (64, 60): 10, (66, 56): 9, (65, 57): 2

Melhor resultado: (64, 60): 10

gen=5000

Tempo de execução: 58.53050318899997 +- 2.873184707884503 s

Melhores resultados: (65, 58): 57, (64, 60): 42, (64, 59): 1

Melhor resultado: (64, 60): 42

O aumento do número de gerações apresentou melhora no resultado final. Entretanto, o aumento de tempo também foi significativo, chegando a quase 1 minuto por teste em gen=5000.

Nos primeiros testes, foi possível observar que os resultados foram bem variados, o que deu indícios de que ainda faltava muito para o AG chegar a um bom resultado.

Já depois de gen=1000, é possível verificar que a quantidade de resultados diferentes diminuem. Nesse teste já é possível encontrar o melhor resultado possível.

7.2 Probabilidade de Mutação (gen=1000, popSize = 20)

mut=0.0

Tempo de execução: 5.498963925470025 +- 0.8513036861816466 s

Melhores resultados: (75, 34): 14, (75, 33): 10, (75, 35): 8, (76, 31): 7, (76, 32): 6, (74, 35): 6, (76, 33): 6, (75, 36): 5, (74, 37): 4, (76, 35): 4, (73, 37): 3, (75, 37): 2, (74, 36): 2, (76, 30): 2, (75, 38): 2, (73, 36): 2, (74, 33): 2, (75, 32): 2, (77, 28): 2, (74, 38): 1, (71, 45): 1, (76, 27): 1, (76, 34): 1, (77, 29): 1, (76, 29): 1, (76, 28): 1, (73, 35): 1, (75, 31): 1, (74, 40): 1, (73, 41): 1

Melhor resultado: (71, 45): 1

mut=0.2

Tempo de execução: 8.270352621690003 +- 1.0198754838213524 s

Melhores resultados: (66, 56): 55, (67, 54): 15, (65, 58): 11, (65, 57): 7, (66, 55): 5, (66, 54): 3, (68, 52): 2, (65, 55): 1, (65, 56): 1

Melhor resultado: (65, 58): 11

mut=0.4

Tempo de execução: 12.060355021749974 +- 1.581207319020692 s

Melhores resultados: (65, 58): 46, (66, 56): 44, (64, 60): 4, (65, 56): 2, (65, 57): 2, (66, 55): 1, (67, 54): 1

Melhor resultado: (64, 60): 4

mut=0.6

Tempo de execução: 13.8768041284 +- 2.1868509857489133 s

Melhores resultados: (65, 58): 71, (66, 56): 22, (64, 60): 3, (64, 59): 2, (65, 57): 2

Melhor resultado: (64, 60): 3

mut=0.8

Tempo de execução: 16.9621103309 +- 2.6949124247374567 s

Melhores resultados: (65, 58): 78, (64, 60): 12, (66, 56): 10

Melhor resultado: (64, 60): 12

mut=1.0

Tempo de execução: 16.9621103309 +- 2.6949124247374567 s

Melhores resultados: (65, 58): 76, (64, 60): 20, (66, 56): 3, (64, 59): 1

Melhor resultado: (64, 60): 20

Foi verificado que a presença da mutação tem um grande impacto positivo nos resultados. Os resultados tendem a ser melhores com altas probabilidades de mutação, tanto pelo resultado em si, quanto pela sua convergência. Contudo, há um gasto de tempo maior quando há maiores chances de mutação, apesar de ele não ser tão alto.

7.3 Tamanho da população (gen=1000, mut=0.4)

popSize = 10

Tempo de execução: 7.276692401319992 +- 1.2638497698967142 s Melhores resultados: (66, 56): 62, (67, 54): 17, (65, 58): 11, (65, 57): 3, (66, 54): 2, (66, 55): 2, (64, 59): 1, (65, 56): 1, (68, 52): 1

Melhor resultado: (64, 59): 1

popSize = 20

Tempo de execução: 12.060355021749974 +- 1.581207319020692 s

Melhores resultados: (65, 58): 46, (66, 56): 44, (64, 60): 4, (65, 56): 2, (65, 57): 2, (66, 55): 1, (67, 54): 1

Melhor resultado: (64, 60): 4

popSize = 30

Tempo de execução: 15.381177004049993 +- 2.4218550387520446 s

Melhores resultados: (65, 58): 66, (66, 56): 27, (65, 57): 4, (64, 60): 3
Melhor resultado: (64, 60): 3

popSize = 50
Tempo de execução: 23.361931398079978 +- 1.8466563521045951 s
Melhores resultados: (65, 58): 78, (66, 56): 12, (64, 60): 8, (64, 59): 1, (64, 58): 1
Melhor resultado: (64, 60): 8

popSize = 70
Tempo de execução: 31.4104342591099988 +- 1.8111549957067574 s
Melhores resultados: (65, 58): 82, (64, 60): 12, (66, 56): 6
Melhor resultado: (64, 60): 12

popSize = 100
Tempo de execução: 44.576567499569975 +- 1.0477707878034745 s
Melhores resultados: (65, 58): 75, (64, 60): 21, (64, 59): 2, (65, 55): 1, (64, 57): 1
Melhor resultado: (64, 60): 21

Houve uma melhora nos resultados com populações maiores, a qual veio acompanhada pelo aumento do tempo e pela convergência dos resultados.

No geral, foi concluído que todos os parâmetros analisados têm um impacto direto nos resultados.