PROJETO 1

Arthur Assis Gonçalves
cc22300@g.unicamp.br

Vinícius Dos Santos Andrade
cc22333@g.unicamp.br

TI327 - Tópicos em Inteligência Artificial
Prof. Dr. Guilherme Macedo
Colégio Técnico de Campinas - UNICAMP

Campinas, 2024

Sumário

1 Introdução **3**

2 Heurística construtiva **3**

2.1 Descrição 4

3 Experimentos computacionais **6**

3.1 Instâncias de testes 6

3.2 Resultados 6

4 Referências Bibliográficas **9**

1 Introdução

O problema que estamos abordando é o conhecido problema do caixeiro viajante, no entanto, com um grau de complexidade adicional. Este é denominado como o problema dos n caixeiros viajantes, que mantém a mesma dinâmica fundamental, mas apresenta uma complexidade adicional devido à necessidade de coordenar e alocar "entregas"entre múltiplos caixeiros.

O problema do caixeiro viajante é um dos problemas mais estudados na área de otimização combinatória e é normalmente é formulado da seguinte maneira: Dado um conjunto de cidades e as distâncias entre elas, qual é o caminho mais curto ao passar em todas as cidades visitando todas elas exatamente uma vez e retorna à cidade de origem?.

No caso do problema dos n caixeiros viajantes, a complexidade é ampliada pelo fato de que há múltiplos caixeiros que precisam ser coordenados para otimizar as rotas e minimizar o custo total.

Neste trabalho, exploraremos uma heurística elaborada por nossa dupla para resolver o problema dos n caixeiros viajantes, como será explicada logo em seguida;

2 Heurística construtiva

A Heurística pensada foi desenvolvida da seguinte forma:

Parâmetros:

- `coords` (lista): Uma lista de coordenadas das cidades. Cada coordenada é uma tupla (x, y).
- `num_caixeiros` (int): O número de caixeiros.

Retorna:

- `rotas` (lista): Uma lista de rotas para cada caixeiro. Cada rota é uma lista de índices das cidades.

2.1 Descrição

A heurística determinística para o problema n-TSP funciona da seguinte maneira:

Definição da Cidade de Origem: A primeira coordenada lida no arquivo é designada como a cidade de origem. Todos os caixeiros iniciam e terminam suas rotas nesta cidade.

Classificação das Cidades: Todas as cidades, exceto a de origem, são classificadas com base em sua distância da cidade de origem. A cidade mais próxima recebe o índice 1, a segunda mais próxima recebe o índice 2, e assim por diante, até a cidade mais distante, que recebe o índice $n - 1$ onde n é o tamanho do vetor de coordenadas lido anteriormente. A cidade de origem é reservada com o índice 0.

Distribuição das Cidades: As cidades são distribuídas igualmente entre os caixeiros. Se o número de cidades não for divisível pelo número de caixeiros, as cidades restantes são distribuídas de forma circular entre eles.

Finalização das Rotas: A cidade de origem é adicionada ao final da rota de cada caixeiro, indicando que todos os caixeiros devem retornar à cidade de origem ao concluir suas rotas.

Número de caixeiros e cidades

```
caixeiros = 3
cidades = 9
```

Coordenadas das cidades

```
coordenadas = [(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
               (6, 6), (7, 7), (8, 8)]
```

A cidade de origem é a cidade 0. As cidades são classificadas e indexadas de acordo com a sua distância da cidade de origem. A cidade 0 recebe o índice 0, a cidade 1 recebe o índice 1, a cidade 2 recebe o índice 2 e assim sucessivamente. A quantidade de cidades são distribuídas igualmente entre os caixeiros. Cada caixeiro recebe índices de cidades de acordo com a sua ordem de classificação.

```
Caixeiro 1: cidades 1, 2, 3
Caixeiro 2: cidades 4, 5, 6
Caixeiro 3: cidades 7, 8
```

O índice da cidade de origem é adicionado ao final de cada rota.

```
rotas = [[0, 1, 2, 3, 0], [0, 4, 5, 6, 0], [0, 7, 8, 0]]
```

Rotas finais para cada caixeiro (Índices das cidades)

```
caixeiro_1 = [0, 1, 2, 3, 0]
caixeiro_2 = [0, 4, 5, 6, 0]
caixeiro_3 = [0, 7, 8, 0]
```

Rotas finais para cada caixeiro (Coordenadas das cidades)

```
coordenadas_caixeiro_1 = [(0, 0), (1, 1), (2, 2), (3, 3), (0, 0)]
coordenadas_caixeiro_2 = [(0, 0), (4, 4), (5, 5), (6, 6), (0, 0)]
coordenadas_caixeiro_3 = [(0, 0), (7, 7), (8, 8), (0, 0)]
```

Distância percorrida por cada caixeiro

```
distancia_caixeiro_1 = 8.485
distancia_caixeiro_2 = 16.9706
distancia_caixeiro_3 = 22.6274
```

Distância total percorrida

```
distancia_total = 48.0833
```

3 Experimentos computacionais

3.1 Instâncias de testes

As instancias utilizadas para teste foram:

Tabela 1: Instancias utilizadas

Teste	Nº cidades	Nº Caixeiros
0	13	1
1	17	1
2	19	1
3	31	3
4	47	3
5	59	3
6	71	5
7	83	5
8	91	5

3.2 Resultados

Tabela 2: 1 caixeiro e 13 cidades

Nº Caixeiro	Rota	Distância Rota
1	[0, 1, 10, 7, 11, 5, 12, 6, 2, 3, 9, 4, 8, 0]	5889.16

Obtivemos uma rota total de: 5889,16.

Tabela 3: 1 Caixeiro e 17 Cidades

Nº	Rota	Distância Rota
1	[0, 2, 11, 16, 14, 3, 13, 15, 7, 9, 1, 12, 8, 4, 6, 10, 5, 0]	9571.629

Obtivemos uma rota total de: 9571,629.

Tabela 4: 1 Caixeiro e 19 Cidades

Nº Caixeiro	**Rota**	**Distância Rota**
1	[0, 17, 10, 11, 16, 2, 14, 18, 15, 9, 5, 8, 13, 7, 4, 6, 1, 3, 12, 0]	11303.97

Obtivemos uma rota total de: 11303,97.

Tabela 5: 3 Caixeiros e 31 Cidades

Nº	**Rota**	**Distância Rota**
1	[0, 4, 28, 5, 24, 21, 23, 19, 22, 25, 7, 31, 0]	4253.987
2	[0, 2, 3, 1, 20, 13, 27, 16, 12, 18, 17, 0]	6103.64
3	[0, 9, 30, 29, 15, 6, 8, 10, 14, 11, 26, 0]	7887.658

Obtivemos uma rota total de: 18245,285.

Tabela 6: 3 Caixeiros e 47 Cidades

Nº	**Rota**	**Distância Rota**
1	[0, 19, 30, 16, 20, 45, 32, 2, 40, 13, 1, 18, 11, 10, 25, 41, 27, 0]	4274.468
2	[0, 42, 28, 22, 7, 23, 15, 34, 26, 39, 14, 43, 47, 31, 35, 9, 36, 0]	8038.442
3	[0, 46, 38, 8, 4, 17, 21, 5, 12, 3, 6, 33, 37, 24, 29, 44, 0]	9485.285

Obtivemos uma rota total de: 21798,196.

Tabela 7: 3 Caixeiros e 59 Cidades

Nº	**Rota**	**Distância**
1	[0, 26, 6, 59, 44, 47, 25, 32, 40, 1, 11, 34, 51, 37, 20, 38, 13, 30, 4, 3, 19, 0]	5593.922
2	[0, 22, 58, 41, 54, 18, 14, 45, 42, 55, 7, 57, 50, 39, 8, 35, 12, 24, 31, 16, 23, 0]	9931.993
3	[0, 5, 17, 49, 27, 43, 9, 33, 29, 56, 46, 15, 2, 36, 28, 53, 48, 52, 10, 21, 0]	10848.071

Obtivemos uma rota total de: 26373.986.

Tabela 8: 5 Caixeiros e 71 Cidades

Nº	**Rota**	**Distância Rota**
1	[0, 32, 28, 8, 42, 24, 53, 7, 27, 65, 13, 1, 48, 34, 64, 57, 0]	5429.882
2	[0, 70, 3, 43, 49, 41, 71, 33, 35, 46, 63, 21, 51, 60, 9, 0]	6331.793
3	[0, 14, 61, 31, 40, 44, 55, 54, 56, 37, 29, 19, 16, 69, 66, 0]	8811.814
4	[0, 50, 20, 10, 18, 62, 12, 68, 22, 45, 38, 59, 58, 30, 67, 0]	6724.79
5	[0, 15, 17, 4, 47, 25, 39, 6, 36, 52, 11, 23, 5, 2, 26, 0]	9285.048

Distância total de todos os viajantes: 36583.327

Tabela 9: 5 Caixeiros e 83 Cidades

Nº	**Rota**	**Distância Rota**
1	[0, 31, 85, 40, 26, 24, 75, 43, 65, 8, 69, 82, 61, 57, 38, 70, 67, 50, 0]	4272.201
2	[0, 60, 59, 51, 83, 46, 9, 11, 13, 77, 37, 41, 21, 39, 23, 4, 12, 62, 0]	6448.231
3	[0, 84, 74, 55, 81, 5, 33, 27, 64, 47, 1, 3, 35, 15, 19, 66, 72, 30, 0]	8495.022
4	[0, 42, 16, 25, 6, 76, 7, 53, 73, 49, 29, 44, 71, 48, 18, 32, 54, 20, 0]	11905.971
5	[0, 17, 2, 63, 56, 58, 45, 10, 22, 78, 80, 14, 28, 52, 34, 36, 79, 68, 0]	14156.958

Distância total de todos os viajantes: 45278.384

Tabela 10: 5 Caixeiros e 91 Cidades

cNº	**Rota**	**Distância**
1	[0, 5, 85, 2, 10, 34, 14, 75, 90, 73, 30, 89, 29, 44, 18, 72, 45, 17, 78, 49, 0]	5971.656
2	[0, 12, 50, 84, 56, 32, 38, 66, 64, 39, 3, 54, 60, 79, 47, 19, 53, 15, 48, 0]	7035.382
3	[0, 33, 87, 55, 7, 81, 24, 46, 40, 35, 61, 22, 63, 83, 76, 74, 37, 88, 23, 0]	8927.666
4	[0, 9, 51, 58, 41, 6, 25, 28, 52, 77, 11, 31, 68, 26, 1, 57, 70, 8, 16, 0]	10006.697
5	[0, 20, 4, 69, 80, 86, 67, 65, 21, 91, 82, 42, 59, 62, 43, 27, 13, 36, 71, 0]	13851.01

A heurística utilizada, apesar de não ótima, atende aos requisitos com extrema rapidez, resolvendo o maior teste em apenas 0.003590 segundos. A distância total percorrida por todos os viajantes foi de 45792.412.

4 Referências Bibliográficas

[1] Miller, C., Tucker, A. e Zemlin, R. A. (1960). Integer Programming Formulations and Traveling Salesman Problems. Journal of the Association for Computing Machinery, 7, 326-329.

[2] Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega, 34. 209-219.