Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática

Modelagem e Otimização Algorítmica - 6903

Relatório de implementação e resultados dos algoritmos Ant Colony e Simulated Annealing

Professor: Igor da Penha Natal

Discente

RA	Nome
112681	Gabriel de Souza Vendrame
115408	Henrique Ribeiro Favaro
112683	Stany Helberth de Souza Gomes da Silva

1. Introdução

Este relatório tem como finalidade analisar a performance dos algoritmos *Ant Colony Optimization* e *Simulated Annealing*. O primeiro sendo aplicado aos problemas do Caixeiro Viajante e Problema da Mochila, já o segundo aplicado apenas ao problema do Caixeiro Viajante.

As soluções foram implementadas na linguagem de programação Python, utilizando bibliotecas para manipulação de vetores *Numpy*.

2. Problemas aplicados

2.1. Problema do caixeiro viajante

Para o problema do caixeiro viajante, dado um conjunto de cidades e as distâncias entre elas, o objetivo é encontrar o caminho mais curto que passe por todas as cidades exatamente uma vez e retorne à cidade inicial. Sabendo que o problema é considerado NP-difícil, não se conhece uma solução algorítmica eficiente para encontrar a solução ótima em tempo polinomial para todas as instâncias do problema.

2.2. Problema da mochila

O Problema da Mochila consiste em determinar a seleção ótima de itens a serem colocados em uma mochila com capacidade limitada, de forma a maximizar o valor total dos itens escolhidos. Dado um conjunto de itens com pesos e valores associados, o objetivo é escolher um subconjunto desses itens que caibam na mochila e possuam o maior valor total possível. Assim como o Problema do Caixeiro Viajante, o Problema da Mochila é considerado NP-difícil e não se conhece uma solução algorítmica eficiente para encontrar a solução ótima em tempo polinomial para todas as instâncias do problema.

3. Resultados obtidos

3.1. Caixeiro Viajante (TSP)

3.1.1. Entrada para o problema

O ATT48 foi utilizado como entrada para o problema do caixeiro viajante. Essa entrada, provida pelo TSPLIB, possui um conjunto de 48 cidades (capitais dos Estados Unidos). A solução ótima para o problema é 33523.

3.1.2. Configuração 1 - Ant Colony (TSP)

- alpha (importância relativa do feromônio): 1.0;
- beta (importância relativa da heurística): 10.0;
- rho (coeficiente residual do feromônio): 0.5;
- q (intensidade do feromônio): 10;
- Número de gerações: 900;
- Número de formigas: 30;
- Critério de parada: atingir o número máximo de gerações.

Execução 1:

```
Custo: 35783.42595280413
Caminho: [3, 25, 9, 41, 47, 4, 28, 1, 33, 40, 15, 21, 2, 22, 10, 11, 32, 45, 14, 39, 8,
 0, 7, 37, 30, 43, 17, 6, 27, 35, 29, 5, 36, 18, 26, 42, 16, 19, 46, 20, 12, 24, 13, 38
, 31, 23, 44, 34]
Gerações: 900
```

Execução 2:

```
Custo: 35454.19815006188
Caminho: [25, 3, 34, 44, 9, 23, 41, 4, 47, 38, 31, 20, 12, 24, 13, 22, 10, 11, 32, 45,
14, 39, 8, 0, 7, 37, 30, 43, 17, 6, 27, 35, 29, 5, 36, 18, 26, 42, 16, 19, 46, 2, 21, 1
5, 40, 33, 28, 1]
Gerações: 900
```

Execução 3:

```
Custo: 36266.11935936634
Caminho: [23, 9, 41, 4, 47, 38, 31, 20, 12, 24, 13, 22, 10, 11, 14, 32, 45, 39, 8, 0, 7
, 37, 30, 43, 17, 6, 27, 35, 29, 5, 36, 18, 26, 42, 16, 19, 46, 33, 2, 21, 15, 40, 28,
1, 25, 3, 34, 44]
Gerações: 900
```

3.1.3. Configuração 2 - Ant Colony (TSP)

- alpha (importância relativa do feromônio): 1,0;
- beta (importância relativa da heurística): 7,0;
- rho (coeficiente residual do feromônio): 0,5;
- q (intensidade do feromônio): 3;
- Número de gerações: 900;
- Número de formigas: 30;
- Critério de parada: atingir o número máximo de gerações.

Execução 1:

```
Custo: 35986.22998651139
Caminho: [33, 40, 15, 21, 2, 22, 10, 11, 32, 45, 14, 39, 8, 0, 7, 37, 30, 43, 17, 6, 27
, 35, 29, 5, 36, 18, 26, 42, 16, 19, 46, 20, 12, 24, 13, 38, 31, 47, 4, 28, 1, 25, 3, 3
4, 44, 9, 23, 41]
Gerações: 900
```

Execução 2:

```
Custo: 35454.19815006187
Caminho: [41, 4, 47, 38, 31, 20, 12, 24, 13, 22, 10, 11, 32, 45, 14, 39, 8, 0, 7, 37, 3
0, 43, 17, 6, 27, 35, 29, 5, 36, 18, 26, 42, 16, 19, 46, 2, 21, 15, 40, 33, 28, 1, 25,
3, 34, 44, 9, 23]
Gerações: 900
```

Execução 3:

```
Custo: 36052.45037851466
Caminho: [46, 10, 22, 13, 24, 12, 20, 38, 31, 47, 4, 28, 1, 25, 3, 34, 44, 9, 23, 41, 3
3, 40, 15, 21, 2, 39, 8, 0, 7, 37, 30, 43, 17, 6, 27, 35, 29, 5, 36, 18, 26, 42, 16, 19
, 32, 45, 14, 11]
Gerações: 900
```

3.1.4. Configuração 1 - Sim. Annealing (TSP)

- Temperatura de parada: 1e-60
- Máximo de iterações: 30000
- alpha: 0,995
- Critério de Parada: Atingir o máximo de iterações ou o máximo de temperatura definido.

Execução 1:

```
Best fitness obtido:  35022.40480131672
Melhoramento sobre a rota aleatória inicial:  14.33%
Iteração de Parada:  27950
Temperatura de Parada:  9.951862821130761e-61
Melhor Solução:  [13, 24, 12, 22, 10, 11, 32, 45, 14, 39, 8, 0, 2, 21, 15, 7, 37, 30
, 43, 17, 6, 35, 27, 5, 36, 18, 26, 16, 42, 29, 19, 46, 20, 31, 38, 47, 4, 41, 23, 9
, 44, 34, 3, 25, 1, 28, 40, 33]
```

Execução 2:

```
Best fitness obtido:  36201.63797228371
Melhoramento sobre a rota aleatória inicial:  7.74%
Iteração de Parada:  27950
Temperatura de Parada:  9.951862821130761e-61
Melhor Solução:  [7, 37, 30, 8, 39, 14, 32, 45, 43, 17, 6, 27, 5, 36, 18, 26, 16, 42,
 29, 35, 19, 46, 11, 10, 22, 13, 24, 12, 20, 31, 38, 47, 4, 28, 41, 23, 9, 44, 34, 3,
 25, 1, 40, 33, 2, 21, 15, 0]
```

Execução 3:

```
Best fitness obtido:  35093.33647135293
Melhoramento sobre a rota aleatória inicial:  14.06%
Iteração de Parada:  27950
Temperatura de Parada:  9.951862821130761e-61
Melhor Solução:  [42, 29, 35, 17, 45, 32, 14, 39, 22, 10, 11, 19, 46, 20, 12, 13, 24,
 4, 47, 38, 31, 23, 44, 34, 3, 25, 9, 41, 1, 28, 40, 33, 2, 21, 15, 0, 7, 8, 37, 30,
43, 6, 27, 5, 36, 18, 26, 16]
```

3.1.5. Configuração 2 - Sim. Annealing (TSP)

- Temperatura de parada: 1e-60
- Máximo de iterações: 70000
- alpha: 0,999
- Critério de Parada: Atingir o máximo de iterações ou o máximo de temperatura definido.

Execução 1:

```
Best fitness obtido:  35036.10648299103
Melhoramento sobre a rota aleatória inicial:  18.05%
Iteração de Parada:  70000
Temperatura de Parada:  2.6621403570873346e-30
Melhor Solução:  [15, 0, 7, 8, 37, 30, 43, 17, 6, 35, 27, 5, 36, 18, 26, 16, 42, 29,
19, 46, 10, 11, 32, 45, 14, 39, 22, 13, 24, 12, 20, 38, 31, 23, 9, 44, 34, 3, 25, 41,
 1, 28, 4, 47, 33, 40, 2, 21]
```

Execução 2:

```
Best fitness obtido:  33784.0270099083
Melhoramento sobre a rota aleatória inicial:  19.51%
Iteração de Parada:  70000
Temperatura de Parada:  2.6621403570873346e-30
Melhor Solução:  [24, 13, 22, 10, 39, 14, 11, 19, 32, 45, 35, 29, 42, 16, 26, 18, 36,
 5, 27, 6, 17, 43, 30, 37, 8, 7, 0, 15, 21, 2, 33, 40, 28, 1, 25, 3, 34, 44, 9, 23, 4
1, 4, 47, 38, 31, 20, 46, 12]
```

Execução 3:

```
Best fitness obtido:  35413.62794282372
Melhoramento sobre a rota aleatória inicial:  9.74%
Iteração de Parada:  70000
Temperatura de Parada:  2.6621403570873346e-30
Melhor Solução:  [26, 18, 36, 5, 27, 6, 17, 43, 30, 37, 7, 0, 8, 39, 21, 15, 2, 33, 4
0, 1, 25, 3, 34, 44, 9, 23, 41, 28, 4, 47, 38, 31, 20, 12, 24, 13, 22, 10, 14, 11, 46
, 19, 32, 45, 35, 29, 42, 16]
```

3.2. Problema da Mochila

3.2.1. Entrada para o problema

Para o problema da mochila, foram gerados valores aleatórios, com a seed 42 de randomicidade, para realização de todos os testes. Mantendo-se essa seed, os valores de entrada nunca serão alterados, porém é possível verificar as diferenças em cada execução quanto à adequação dos itens escolhidos.

Os itens contam com valores, que são somados quando adicionados à mochila, resultando num maior valor de importância, e um peso, que pode ou não caber dentro da capacidade da mochila conforme outros são adicionados.

3.2.2. Configuração 1 - Ant Colony (Mochila)

- alpha: 1;
- beta: 1;
- Número de formigas: 20;
- Máximo de iterações: 300;
- taxa de evaporação: 0.2;
- Capacidade da mochila: 370 (“kg”);
- Critério de parada: atingir o número máximo de gerações definido.

Execução 1:

```
Melhor arranjo de itens: [0, 4, 6, 11, 17, 24, 34, 37, 47, 55, 56, 58, 83, 133, 137,
146, 148, 158, 160, 165, 178, 189, 190, 201, 216, 229, 231, 236, 238, 239, 244, 247,
253, 264, 267, 272, 275, 280, 284]
Melhor valor alcançado: 2558.0
Peso total utilizado: 369
Critério de parada utilizado: Máximo de 300 iterações
```

Execução 2:

```
Melhor arranjo de itens: [0, 4, 6, 11, 17, 24, 34, 55, 56, 83, 105, 123, 133, 137, 14
6, 148, 158, 160, 165, 178, 189, 190, 201, 216, 229, 231, 236, 238, 239, 241, 244, 24
7, 253, 264, 267, 272, 275, 280, 284]
Melhor valor alcançado: 2563.0
Peso total utilizado: 370
Critério de parada utilizado: Máximo de 300 iterações
```

Execução 3:

```
Melhor arranjo de itens: [0, 4, 6, 11, 17, 18, 34, 55, 56, 83, 123, 133, 137, 146, 14
8, 158, 160, 165, 178, 189, 190, 201, 216, 229, 231, 236, 238, 239, 241, 244, 247, 25
3, 264, 267, 272, 275, 280, 284]
Melhor valor alcançado: 2581.0
Peso total utilizado: 370
Critério de parada utilizado: Máximo de 300 iterações
```

3.2.3. Configuração 2 - Ant Colony (Mochila)

- alpha: 0.3;
- beta: 1;
- Número de formigas: 30;
- Máximo de iterações: 300;
- taxa de evaporação: 0.5;
- Capacidade da mochila: 370 (“kg”);
- Critério de parada: atingir o número máximo de gerações definido.

Execução 1:

```
Melhor arranjo de itens: [0, 4, 6, 11, 12, 17, 18, 34, 55, 56, 83, 85, 133, 146, 148,
 158, 160, 165, 181, 189, 190, 201, 216, 231, 236, 239, 242, 244, 253, 264, 265, 272,
 275, 284]
Melhor valor alcançado: 2406.0
Peso total utilizado: 370
Critério de parada utilizado: Máximo de 300 iterações
```

Execução 2:

```
Melhor arranjo de itens: [0, 11, 17, 18, 33, 55, 56, 83, 105, 113, 137, 146, 148, 158
, 160, 162, 178, 189, 190, 201, 216, 229, 231, 236, 238, 239, 241, 244, 247, 253, 264
, 272, 275, 280, 284]
Melhor valor alcançado: 2343.0
Peso total utilizado: 369
Critério de parada utilizado: Máximo de 300 iterações
```

Execução 3:

```
Melhor arranjo de itens: [0, 4, 6, 11, 17, 34, 55, 58, 83, 87, 118, 128, 146, 148, 16
0, 165, 178, 189, 190, 201, 216, 229, 236, 238, 239, 241, 244, 247, 253, 264, 272, 27
5, 280, 284, 288]
Melhor valor alcançado: 2406.0
Peso total utilizado: 367
Critério de parada utilizado: Máximo de 300 iterações
```

3.3. Análise dos resultados

É possível notar, para o problema do caixeiro viajante aplicado ao *Ant Colony*, que ambas as configurações não apresentaram grandes diferenças, chegando a resultados similares e ainda longe do ótimo esperado.

Já quanto ao *Simmulated Annealing*, este apresentou resultados melhores para a configuração 2, quando tem-se um aumento quantidade de iterações possíveis, junto ao incremento do alpha, o que permite uma maior demora na limitação da temperatura, fazendo com que um maior número de soluções possa ser verificado até diminuir-se o valor para o *best fitness*.

Já para o problema da mochila, aplicado ao *Ant Colony*, é possível observar que os resultados obtidos quase sempre preenchem por completo o peso disponível na mochila, porém variando-se o valor total de importância dos itens adicionados. Para a configuração 1, resultados melhores foram obtidos quando considera-se o valor de importância do arranjo, o que contempla melhor o escopo do problema. Na configuração 2, ainda que utilizando-se do mesmo peso disponível e com 300 iterações de limitação, por conta da redução de alpha, que seria a importância do feromônio, e, provavelmente pelo aumento da taxa de evaporação do feromônio, os resultados obtidos foram piores do que os anteriores pela outra configuração.

3.4. Comparação dos Resultados

Com base nos resultados obtidos, pode-se realizar uma comparação com os resultados obtidos anteriormente pelo Algoritmo Genético.

Nota-se pelos resultados obtidos no Algoritmo Genético uma melhor solução atingida de aproximadamente 33.588, na primeira execução da segunda configuração. Já para o algoritmo Ant Colony, encontra-se um valor de 35.454 em sua execução 2°, tanto da primeira quanto da segunda configuração, demonstrando uma menor performance em relação ao algoritmo genético.

No cenário do problema estudado, para o caixeiro viajante, executado em ambos os algoritmos, nota-se essa grande diferença no resultado final, onde o genético pode se mostrar mais eficaz, visto que aumentar a quantidade de gerações do Ant Colony, ainda que pudesse melhorar o resultado, acabaria gerando um maior custo computacional do que o genético, de maneira que utilizou-se um número de gerações que não demanda grande quantidade de processamento, apenas para fins de comparação.

Conclui-se que, dependendo da configuração e ajuste dos parâmetros, ambos os algoritmos podem se aproximar da solução ótima esperada, porém os custos de execução para cada um podem variar de diferentes maneiras, fazendo com que, no presente estudo, o algoritmo genético fosse escolhido como melhor opção para obtenção dos resultados com menor custo.