Análise do Problema do Caixeiro Viajante com Lucro Utilizando as Metaheurísticas *Tabu Search* e *Simulated Annealing*

Diego Américo Guedes
diegoamericoguedes@gmail.com
Instituto de Informática – Universidade Federal de Goiás

***Resumo* — Este é um relatório técnico que mostra um comparativo entre as metaheurísticas *Tabu Search* e *Simulated Annealing utilizando* o problema do Caixeiro viajante com lucro.**

Palavras-chave—metaheurísticas, caixeiro viajante com lucro, tabu search, simulated annealing

I. INTRODUÇÃO

O problema do caixeiro viajante (*Traveling Salesman Problem - TSP*) é um problema de otimização que visa visitar todas as cidades definidas no problema com o menor custo possível. Uma variante desse problema, chamado problema do caixeiro viajante com lucros (T*raveling Salesman Problem with Profits - TSPP)* é visitar a maior quantidade possíveis de cidades com o menor custo possível. A medida que aumenta-se a complexidade do problema, que é aumentando o número de cidades, uma solução determinística mostra-se impraticável por o TSPP ser um problema NP-Difícil [1]. Por isso, este relatório se propõe a fazer uma análise da solução do TSPP utilizando as metaheurísticas *Tabu Search* e *Simulated Annealing* que, apesar de nem sempre apresentarem a solução ótima, os resultados podem mostrar-se bem satisfatórios e computacionalmente viáveis.

A Tabela Tabu (*Tabu Search* – TS) é uma metaheurística para guiar uma busca local usando estruturas de memória [2]. A partir de uma solução inicial, efetua-se movimentos para determinados vizinhos, exceto os vizinhos que estão na lista chamada lista tabu. Na Figura 1, pode-se observar o pseudocódigo do algoritmo TS.

Algoritmo 1 Busca Tabu

1: **Entrada**: $f(.), N(.), |T|, |V|, s$
2: **Saída**: s^*
3: $s^* \leftarrow s$ {Melhor solução encontrada até então}
4: $Iter \leftarrow 0$ {Contador do número de iterações}
5: $MelhorIter \leftarrow 0$ {Iteração mais recente que forneceu s^*}
6: $T \leftarrow \emptyset$ {lista tabu}
7: **enquanto** Critério de parada não satisfeito **faça**
8: $Iter \leftarrow Iter + 1$
9: Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subseteq N(s)$ tal que o movimento m não seja tabu ($m \notin T$) ou s' atenda a um critério de aspiração
10: Atualize a lista tabu T
11: $s \leftarrow s'$
12: **se** $f(s) < f(s^*)$ **então**
13: $s^* \leftarrow s$
14: $MelhorIter \leftarrow Iter$
15: **fim se**
16: **fim enquanto**
17: Retorne s^*

Fig. 1. Pseudocódigo do algoritmo TS [2]

O Recozimento Simulado (*Simulated Annealing* – SA*)* é uma metaheurística que busca a vizinhança utilizando uma técnica baseada no processo de resfriamento de materiais utilizado nos processos metalúrgicos. O algoritmo utiliza o critério de Metrópoles para simular o equilíbrio térmico [3]. Uma das principais característica do SA é a aceitação de soluções de piora do valor da função objetivo, com uma probabilidade decrescente ao longo das iterações, que é decaimento da temperatura, a fim de escapar de ótimos locais. Na Figura 2, pode-se observar o seu pseudocódigo do algoritmo SA.

Select an initial solution s_0;
Select an initial temperature $t_0 > 0$;
Select a temperature reduction function α;
Repeat
  Repeat
    Randomly select $s \in N(s_0)$;
    $\delta = f(s) - f(s_0)$;
    If $\delta < 0$
    then $s_0 = s$
    else
    generate random x uniformly in the range $(0, 1)$;
      if $x < \exp(-\delta/t)$ then $s_0 = s$;
  Until $iteration\_count = nrep$
  Set $t = \alpha(t)$;
Until stopping condition = true.
s_0 is the approximation to the optimal solution.

Fig. 2. Pseudocódigo do algoritmo SA [4]

II. RESULTADOS

A. Parâmetros

Os códigos fontes e arquivos de testes encontram-se no github [5]. O códigos fontes foram escritos na linguagem de programação python utilizando a plataforma Colab do Google [6]. Foram utilizados dois arquivos de testes [5], dataset-HP.xls e dataset-LP.xls, que contém o custo de visitar a respectiva cidade. O que os diferencia é que os pesos em dataset-HP.xls são maiores que os pesos em dataset-LP.xls.

Os parâmetros utilizados na TS, conforme tabela 1, foram uma Lista Tabu de tamanho variando de 5 a 25, escolhida de forma aleatória, e 1000 iterações da busca tabu.

TABELA 1. PARÂMETROS DA METAHEURÍSTICA TABU SEARCH

Parâmetros	Valor
Tamanho da tabela Tabu	5 - 25[a]
Quantidade Iterações	1000

[a] Os valores foram escolhidos de forma aleatória

Os parâmetros utilizados no SA, conforme tabela 2, foram uma Temperatura inicial de 1, temperatura mínima de 0,0001 e fator de redução de temperatura de 0,9.

TABELA 2. PARÂMETROS DA METAHEURÍSTICA SIMULATED ANNEALING

Parâmetros	Valor
Temperatura Inicial	1
Temperatura Mínima	0,0001
Fator de redução de temperatura	0,9

B. *Análise dos resultados*

Por se tratarem de metaheurísticas, algoritmos não determinísticos, os testes foram executados 10 vezes e os resultados apresentados a seguir serão a média aritmética das 10 execuções.

A tabela 3 apresenta os resultados da execução das duas metaheurísticas utilizando com o parâmetro de 51, 76 e 101 clientes. Observa-se que a função objetivo e quantidade de clientes visitados têm resultados muito semelhantes nas duas metaheurísticas, ou seja, a diferença entres os resultados não é significativa. No entanto, a diferença do tempo de processamento é bastante expressiva, sendo a TS por volta de 4 vezes mais rápido que o SA. Assim, para o problema e base de dados utilizadas, a TS mostra-se uma melhor escolha por ter função objetivo muito semelhante ao AS, mas com tempo de processamento bem inferior.

TABELA 3. COMPARATIVO DAS METAHEURÍSTICAS TABU SEARCH E SIMULATED ANNEALING – BASE DE DADOS HP

Parâmetros	*Tabu Search*	*Simulated Annealing*
51 Clientes		
Valor da Função Objetivo	699,33	697,85
Quantidade de Clientes Visitados	49,1	49,0
Tempo de Processamento (segundos)	4,99	18,17
76 Clientes		
Valor da Função Objetivo	1231.70	1229.93
Quantidade de Clientes Visitados	73.7	74.3
Tempo de Processamento (segundos)	14.02	63.15
101 Clientes		
Valor da Função Objetivo	1621.86	1622.41
Quantidade de Clientes Visitados	99.4	99.1
Tempo de Processamento (segundos)	38.95	162.33

Os resultados das análises da base de dados LP com o parâmetro 51, 76 e 101 clientes encontram-se na tabela 4. Observa-se que a função objetivo e quantidade de clientes visitados têm uma diferença mais relevante entre as metaheurísticas. Ademais, o tempo de processamento do TS continua bem inferior ao SA. Por isso, para o problema e base de dados utilizadas, a TS continua a melhor escolha.

TABELA 4. COMPARATIVO DAS METAHEURÍSTICAS TABU SEARCH E SIMULATED ANNEALING – BASE DE DADOS LP

Parâmetros	*Tabu Search*	*Simulated Annealing*
51 Clientes		
Valor da Função Objetivo	42.92	29.52
Quantidade de Clientes Visitados	22.6	29.9
Tempo de Processamento (segundos)	2.69	5.35
76 Clientes		
Valor da Função Objetivo	141.93	133.59
Quantidade de Clientes Visitados	57.2	62.4
Tempo de Processamento (segundos)	12.47	38.99
101 Clientes		
Valor da Função Objetivo	246.93	241.86
Quantidade de Clientes Visitados	76.1	80.8
Tempo de Processamento (segundos)	29.70	109.01

CONCLUSÃO

Os resultados encontrados mostram que, apesar da complexidade do problema do caixeiro viajante com lucro, é possível resolvê-lo de forma satisfatório em um tempo computacional aceitável utilizando as metaheurística *Tabu Search* e *Simulated Annealing*. Para as bases de dados utilizadas nos testes, o *Tabu Search* mostrou-se a melhor metaheurística a ser utilizada, pois possui bons resultados em sua função objetivo, além de um menor custo computacional.

REFERÊNCIAS

[1] Dominique Feillet, Pierre Dejax, Michel Gendreau, "TRAVELING SALESMAN PROBLEMS WITH PROFITS: AN OVERVIEW," ORP , SEPTEMBER 26-29 2001.

[2] Marconde J. F. Souza, "Manual de computação evolutiva e metaheurística", Capítulo 8 – Busca Tabu, Universidade de Coimbra, 2012

[3] Aarst, E. Korst, J, E. Simulated Annealing and Boltzmann Machines – a stochastica approach to combinatorial optimization and neural computing, Essex: Jonh Wiley & Sons, 1989.

[4] Dowsland, K. A. Simulated Annealing. In REEVES, C.R. (Ed.) Modern heuristic techniques for combinatorial problems, London: McGraw-Hill, 1995

[5] Diego Américo Guedes, Github – Hospedagem de código-fonte, Jul. 2020. Acessado em: Jul. 10, 2020. [Online]. Disponível: https://github.com/diegoguedes/metaheuristicas

[6] Google, Colab – Texto com códigos executáveis, Jul. 2020. Acessado em: Jul. 10, 2020. [Online]. Disponível: https://colab.research.google.com/