Heurística Gulosa Construtiva para o TSP

Vitor dos Santos Amorim[1]

***Abstract*— Exercício entregue na disciplina de Otimização Combinatória e Metaheurísticas como forma parcial no processo de avaliação.**

I. QUESTÃO

1) Em anexo, duas instâncias do problema do caixeiro viajante (TSP). O formato é *vértice* $coordenada_x$ $coordenada_y$.

   Queremos achar soluções heurísticas para essa instância.

   Implementar a Heurística Gulosa Construtiva baseada em:

   a) Vizinho mais próximo (execute a heurística 10 vezes, iniciando de 10 pontos aleatoriamente escolhidos. Selecione o melhor resultado das 10 execuções da heurística).
   b) Para iniciar, monte um circuito com 3 nós: Escolha o nó mais próximo ao um circuito/ Introduza o nó escolhido no circuito na melhor posição usando o cálculo, conforma a Equação 1.

$$\delta(k)_{ij} = C_{ik} + C_{jk} - C_{ij} \tag{1}$$

2) 2. Compare os resultados em (a) e (b) em termos de tempo de execução e qualidade da solução.

Resolução

1. Foi utilizado uma heurística gulosa construtiva, baseada na inserção mais econômica, para resolver os dois problemas de otimização para o problema do caixeiro viajante, o primeiro contendo 395 cidades, e o segundo com 734. O programa foi implementado em linguagem de programação *Python*, com o compilador *Python* 3.8, num ambiente cuja máquina possuia um processador da Intel Core i5-250K CPU @ 3.30GHz, Memória RAM de 8GB, no sistema operacional Ubuntu 20.04.5 LTS.

O ponto inicial para cada uma das 10 rodadas foi escolhido com base na função aleatória *random.randint(1, len(points))*, ou seja, foram escolhidos pontos entre 1 e o número máximo de coordenadas do arquivo. E para cada uma das iterações, foi definida a *seed* da função *random*, como sendo respectivamente os valores da lista, $[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$, permitindo assim que os resultados sejam replicados e testados.

Arquivo	Dimensão	Distância	$t(min)$
pbl395.tsp	395	1536.17	9.729
uy734.tsp	734	94338.47	70.1413

TABLE I: Comparação entre o melhor resultados dos dois conjuntos de dados.

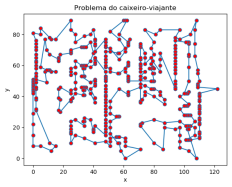
TSP	**Min.**	**Max.**	$\bar{x}$	σ	**Tempo(min)**
395	1536.17	1578.45	1557.374	12.07	9.73
734	94338.47	95951.48	95204.73	570.51	70.14

TABLE II: Resultados Obtidos em 10 rodadas de teste.

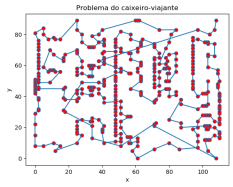
Foi elaborado também, o algoritmo guloso para resolver o problema do caixeiro viajante, porém diferente do primeiro caso, a inserção da nova coordenada no circuito era obtido calculando-se qual ponto do circuito estava mais próximo ao último ponto adicionado. O resultado do mesmo está representado pela Figura 1b.

Heurística Gulosa	Melhor Solução	$t(min)$
Vizinho mais próximo	1547.56	0.038
Inserção mais barata	1536.17	9.729

TABLE III: Analise com base no arquivo de dados: pbl395.tsp.



(a) Melhor resultado da heurística gulosa construtiva baseada na inserção mais barata.



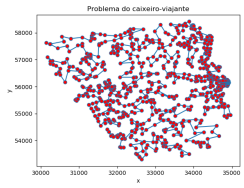
(b) Melhor resultado da heurística gulosa baseada no vizinho mais próximo.

Fig. 1: Comparação entre as diferentes formas de construção do circuito.

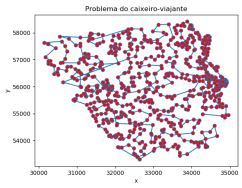
De forma análoga, a mesma análise foi realizada para o caso com 734 pontos e o resultado obtido é apresentado pela Figura 2.

Heurística Gulosa	Melhor Solução	$t(min)$
Vizinho mais próximo	96814.32	0.0836
Inserção mais barata	94338.47	70.1413

TABLE IV: Analise com base no arquivo de dados: uy734.tsp.



(a) Melhor resultado da heurística gulosa construtiva baseada na inserção mais barata.



(b) Melhor resultado da heurística gulosa baseada no vizinho mais próximo.

Fig. 2: Comparação entre as diferentes formas de construção do circuito.

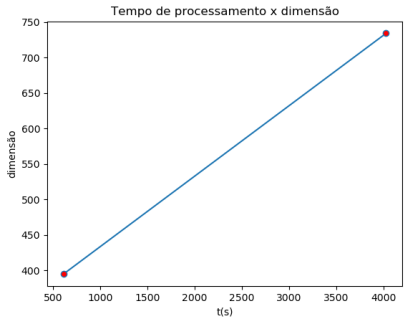


Fig. 3: Tempo de processamento versus dimensão.

A. *Conclusão*

Conforme esperado, a heurística de construção gulosa com inserção mais barata apresentou um melhor resultado, reduzindo o percurso total percorrido em 2.56% quando comparada com o outro método para o caso com 734 coordenadas, todavia o custo computacional aumentou em $8.36 * 10^4\%$. Portanto é importante avaliar os recursos computacionais disponíveis, o quão boa precisa ser a solução encontrada, e recomenda-se como trabalhos futuros encontrar formas de processar e analisar a informação de modo a reduzir o esforço computacional dispendido.