Trabalho Prático 2 - Algoritmos 2

Milena Corrêa Moreira

December 11, 2023

1 Introdução

O presente trabalho tem por objetivo analisar aspectos práticos de algoritmos para a solução de problemas difíceis. Em especial, serão avaliadas as implementações dos algoritmos para computação de rotas no problema do caixeiro viajante. Primeiro será avaliada uma solução exata, baseada em branch-and-bound, e depois serão avaliadas duas soluções aproximadas, sendo elas o algoritmo Twice-Around-The-Tree e o algoritmo de Christofides.

Os algoritmos serão testados com instâncias simétricas do problema do caixeiro viajante, obtidas a partir da biblioteca TSPLIB 95.

Este trabalho foi desenvolvido em Python. As instâncias do problema foram representadas por grafos completos e ponderados. Para implementar estes grafos, foi utilizada a biblioteca Networkx.

Para cada instância, temos um conjunto de cidades, onde cada cidade é representada por duas coordenadas no plano. Cada cidade foi representada por um nó no grafo, e a distância euclidiana entre as coordenadas de cada cidade é o peso entre dois nós no grafo.

2 O problema do caixeiro viajante

O problema do caixeiro-viajante (TSP) é um problema de otimização combinatória que tenta determinar a menor rota para percorrer uma série de cidades, visitando cada uma delas exatamente uma vez e retornando à cidade de origem. O TSP é um problema NP-difícil, o que significa que não há algoritmo conhecido que possa resolvê-lo em tempo polinomial. O TSP tem aplicações em diversas áreas, como logística, transporte, planejamento de rotas, design de circuitos, entre outras. Por exemplo, empresas de entrega de encomendas podem usar o TSP para otimizar as rotas de seus motoristas, reduzindo o tempo e os custos de transporte.

Neste trabalho, todas as instâncias do TSP testadas tem custo ótimo conhecido, e a qualidade de cada solução aproximada foi definida como a divisão do tamanho da solução aproximada pelo tamanho da solução ótima.

3 Branch and Bound

O algoritmo branch and bound é um método de solução exata para o problema do caixeiro-viajante. Ele é baseado em uma estratégia onde o espaço de busca é dividido em subespaços menores e mais gerenciáveis, que são explorados de forma sistemática. O algoritmo começa com uma solução inicial e, em seguida, divide o espaço de busca em subespaços menores, descartando subespaços que não contêm soluções melhores do que a melhor solução atual. O processo é repetido até que todas as soluções possíveis tenham sido exploradas e a melhor solução seja encontrada.

Este algoritmo possui uma complexidade de tempo exponencial, sendo, portanto, difícil aplicá-lo para instâncias do TSP.

Neste trabalho, a menor instância do TSP tem 51 cidades. Verificou-se para esta instância que o algoritmo não é capaz de chegar a uma solução ótima em tempo hábil. Por conseguinte, foi possível generalizar que a execução do branch and bound não é viável para as instâncias do TSP que têm um número de cidades maior do que 51. Assim, embora tenha havido tentativas de executar a abordagem branch and bound para algumas instâncias da biblioteca TSPLIB 95, não foi possível obter respostas.

Tendo isso em vista, fica nítida a importância dos algoritmos aproximativos para problemas NP-Difíceis como o TSP.

4 Twice around the tree

O algoritmo twice around the tree é um algoritmo polinomial 2-aproximado para o problema do caixeiro-viajante. Ele é baseado em uma estratégia de construir um circuito euleriano a partir de uma árvore geradora mínima (MST) e, em seguida, transformar o circuito em um ciclo hamiltoniano. O algoritmo começa com a construção de uma MST a partir do grafo original. Em seguida, é criado um circuito euleriano a partir da MST, duplicando cada aresta da MST. Finalmente, o circuito euleriano é transformado em um ciclo hamiltoniano, eliminando as repetições de vértices.

Neste trabalho, o algoritmo utilizado para gerar a MST foi o algoritmo de Prim. Já o circuito euleriano foi obtido a partir do algoritmo de busca em profundidade.

Este algoritmo é uma heurística simples e eficaz para o TSP, embora não forneça garantias de otimalidade. Pode ser uma escolha razoável para instâncias em que a eficiência computacional é mais crucial do que a garantia de obter uma solução ótima. Uma vez que ele é 2-aproximado, o comprimento da solução encontrada pelo algoritmo será no máximo 2 vezes o comprimento da solução ótima.

5 Algoritmo de Christofides

O algoritmo de Christofides é um algoritmo polinomial 1,5-aproximado para o problema do caixeiro-viajante. Ele é aplicável nos casos em que as distâncias entre as cidades formam um espaço métrico, ou seja, são simétricas e obedecem à desigualdade triangular.

O algoritmo de Christofides começa com a construção de uma árvore geradora mínima (MST) a partir do grafo original. Em seguida, o algoritmo encontra um conjunto de vértices de grau ímpar na MST e constrói um emparelhamento perfeito de peso mínimo nesse conjunto. O algoritmo então combina as arestas da MST e do emparelhamento perfeito para formar um multigrafo euleriano, que é um grafo onde cada vértice tem grau par. O algoritmo então encontra um circuito euleriano no multigrafo euleriano e transforma o circuito em um ciclo hamiltoniano, eliminando as repetições de vértices.

O Algoritmo de Christofides é uma heurística eficiente que oferece uma garantia teórica de desempenho em relação à solução ótima. A razão de aproximação de 1,5 faz do algoritmo de Christofides uma escolha popular para instâncias do TSP em que uma solução aproximada de boa qualidade é suficiente.

6 Conclusão

A partir da execução das instâncias simétricas do TSP, foi possível perceber que os algoritmos aproximativos são uma alternativa eficiente e razoável para o problema do caixeiro viajante quando se trata de instâncias de tamanho mediano. Embora tenham sido incapazes de obter a solução ótima para todas as instâncias testadas, o tamanho das soluções encontradas pelo algoritmo twice around the tree foram, em média, 1,38 vezes maiores do que as soluções ótimas. Já o algoritmo de Christofides obteve soluções que foram em média 1,11 vezes maiores do que as soluções ótimas.

A abordagem branch and bound se mostrou ineficiente para as menores instâncias, embora seja a única implementação dentre as três apresentadas neste trabalho que obtenha a solução ótima.

Abaixo estão três tabelas contendo os resultados da execução dos algoritmos twice around the tree e Christofides para instâncias do problema do caixeiro viajante. As instâncias estão ordenadas de acordo com o número de nós.

Ao longo da tabela é possível analisar que embora o algoritmo de Christofides tenha obtido resultados melhores do que o algoritmo twice around the tree, este segundo teve a vantagem de ser muito mais rápido para instâncias maiores do problema.

Para simplificar os testes, todas as execuções que passaram dos trinta minutos foram interrompidas. Nestes casos, o resultado do algoritmo aproximativo não foi reportado.

Dataset	Algoritmo	Limiar	Resultado	Aproximação	Tempo (segundos)
eil51	twice around the tree	426	626.34	1.47	0.0047
eil51	christofides	426	486.82	1.14	0.0416
berlin52	twice around the tree	7542	10116.01	1.34	0.0047
berlin52	christofides	7542	8594.03	1.14	0.0397
st70	twice around the tree	675	873.35	1.29	0.0074
st70	christofides	675	759.16	1.12	0.1032
eil76	twice around the tree	538	703.64	1.31	0.0093
eil76	christofides	538	623.06	1.16	0.1402
pr76	twice around the tree	108159	145338.11	1.34	0.0094
pr76	christofides	108159	116683.53	1.08	0.0626
rat99	twice around the tree	1211	1717.28	1.42	0.0140
rat99	christofides	1211	1367.43	1.13	0.1552
kroA100	twice around the tree	21282	27211.68	1.28	0.0283
kroA100	christofides	21282	23292.98	1.09	0.2020
kroB100	twice around the tree	22141	25881.20	1.17	0.0143
kroB100	christofides	22141	24009.19	1.08	0.1393
kroC100	twice around the tree	20749	27966.54	1.35	0.0146
kroC100	christofides	20749	23469.13	1.13	0.1933
kroD100	twice around the tree	21294	27113.30	1.27	0.0146
kroD100	christofides	21294	23587.75	1.11	0.1898
kroE100	twice around the tree	22068	30507.42	1.38	0.0147
kroE100	christofides	22068	23782.61	1.08	0.2599
rd100	twice around the tree	7910	10791.66	1.36	0.0149
rd100	christofides	7910	8905.49	1.13	0.3144
eil101	twice around the tree	629	832.22	1.32	0.0152
eil101	christofides	629	723.00	1.15	0.2810
lin105	twice around the tree	14379	20044.57	1.39	0.0407
lin105	christofides	14379	16323.68	1.14	0.1861
pr107	twice around the tree	44303	54238.04	1.22	0.0196
pr107	christofides	44303	47894.18	1.08	0.1371
pr124	twice around the tree	59030	75177.54	1.27	0.0253
pr124	christofides	59030	64437.70	1.09	0.1160
bier127	twice around the tree	118282	157605.76	1.33	0.0334
bier127	christofides	118282	132456.25	1.12	0.5109
ch130	twice around the tree	6110	8128.76	1.33	0.0241
ch130	christofides	6110	6752.28	1.11	0.2740
pr136	twice around the tree	96772	151913.74	1.57	0.0273
pr136	christofides	96772	103771.91	1.07	0.1051
pr144	twice around the tree	58537	80596.32	1.38	0.0320
pr144	christofides	58537	70405.28	1.20	0.1060
ch150	twice around the tree	6528	8333.47	1.28	0.0310
ch150	christofides	6528	7112.04	1.09	0.3126
kroA150	twice around the tree	26524	35122.57	1.32	0.0311
kroA150	christofides	26524	29688.32	1.12	0.6940
kroB150	twice around the tree	26130	36154.73	1.38	0.0332
kroB150	christofides	26130	30054.39	1.15	0.7546
pr152	twice around the tree	73682	92021.71	1.25	0.0398
pr152	christofides	73682	79314.35	1.08	0.1290
u159	twice around the tree	42080	58292.51	1.39	0.0368
u159	christofides	42080	47581.72	1.13	0.4257
rat195	twice around the tree	2323	3329.67	1.43	0.0553
rat195	christofides	2323	2648.10	1.14	0.8032
d198	twice around the tree	15780	19161.21	1.21	0.0794
d198	christofides	15780	17316.37	1.10	0.8996

Table 1: Tabela com dados de execução das instâncias do TSP.

Dataset	Algoritmo	Limiar	Resultado	Aproximação	Tempo (segundos)
kroA200	twice around the tree	29368	40030.87	1.36	0.0586
kroA200	christofides	29368	33602.86	1.14	1.4607
kroB200	twice around the tree	29437	40710.96	1.38	0.0580
kroB200	christofides	29437	33118.12	1.13	1.1359
ts225	twice around the tree	126643	196979.28	1.56	0.0719
ts225	christofides	126643	133282.17	1.05	0.2322
tsp225	twice around the tree	3919	5134.28	1.31	0.0742
tsp225	christofides	3919	4376.00	1.12	1.0632
pr226	twice around the tree	80369	115264.01	1.43	0.0797
pr226	christofides	80369	92425.75	1.15	0.8099
gil262	twice around the tree	2378	3358.15	1.41	0.1059
gil262	christofides	2378	2695.15	1.13	2.3941
pr264	twice around the tree	49135	70613.79	1.44	0.1278
pr264	christofides	49135	54663.60	1.11	0.8101
a280	twice around the tree	2579	3751.40	1.45	0.1517
a280	christofides	2579	2975.11	1.15	1.7945
pr299	twice around the tree	48191	64573.23	1.34	0.1354
pr299	christofides	48191	52968.22	1.10	2.5175
lin318	twice around the tree	42029	59334.95	1.41	0.1935
lin318	christofides	42029	47259.76	1.12	3.2361
rd400	twice around the tree	15281	20250.38	1.33	0.2731
rd400	christofides	15281	17551.75	1.15	9.8838
fl417	twice around the tree	11861	16299.73	1.37	0.3204
fl417	christofides	11861	13424.73	1.13	3.8749
pr439	twice around the tree	107217	146846.65	1.37	0.3950
pr439	christofides	107217	119532.04	1.11	4.0962
pcb442	twice around the tree	50778	69762.40	1.37	0.3297
pcb442	christofides	50778	54876.77	1.08	7.6138
d493	twice around the tree	35002	45571.72	1.30	0.4818
d493	christofides	35002	38393.64	1.10	12.5976
u574	twice around the tree	36905	49086.37	1.33	0.6676
u574	christofides	36905	41337.61	1.12	20.3299
rat575	twice around the tree	6773	9425.89	1.39	0.6389
rat575	christofides	6773	7802.75	1.15	22.9955
p654	twice around the tree	34643	49620.11	1.43	0.9217
p654	christofides	34643	39244.71	1.13	3.6994
d657	twice around the tree	48912	65730.19	1.34	0.8377
d657	christofides	48912	54738.30	1.12	30.2490
u724	twice around the tree	41910	57934.09	1.38	1.1250
u724	christofides	41910	47954.68	1.14	38.4209
rat783	twice around the tree	8806	12067.60	1.37	1.2602
rat783	christofides	8806	10102.87	1.15	57.7003
pr1002	twice around the tree	259045	351387.22	1.36	2.2709
pr1002	christofides	259045	286233.10	1.10	87.7256
u1060	twice around the tree	224094	303084.36	1.35	2.8268
u1060	christofides	224094	249108.58	1.11	118.8171
vm1084	twice around the tree	239297	314321.66	1.31	2.9773
vm1084	christofides	239297	260833.06	1.09	61.0738
pcb1173	twice around the tree	56892	80728.19	1.42	3.4991
pcb1173	christofides	56892	63392.44	1.11	108.4330
d1291	twice around the tree	50801	80210.59	1.58	4.2665
d1291	christofides	50801	57930.37	1.14	30.3211
rl1304	twice around the tree	252948	350894.10	1.39	4.7177
rl1304	christofides	252948	278319.19	1.10	30.2818
rl1323	twice around the tree	270199	384134.45	1.42	4.9686
rl1323	christofides	270199	298261.72	1.10	31.5189

Table 2: Tabela com dados de execução das instâncias do TSP.

Dataset	Algoritmo	Limiar	Resultado	Aproximação	Tempo (segundos)
nrw1379	twice around the tree	56638	79156.27	1.40	4.6553
nrw1379	christofides	56638	63845.07	1.13	383.5674
fl1400	twice around the tree	20127	27825.07	1.38	6.0986
fl1400	christofides	20127	22834.35	1.13	301.3017
u1432	twice around the tree	152970	222340.34	1.45	6.4256
u1432	christofides	152970	171640.92	1.12	101.3936
fl1577	twice around the tree	22249	32026.55	1.44	7.9335
fl1577	christofides	22249	24515.85	1.10	72.9762
d1655	twice around the tree	62128	87803.08	1.41	7.8566
d1655	christofides	62128	70555.34	1.14	121.1790
vm1748	twice around the tree	336556	445935.55	1.32	9.4490
vm1748	christofides	336556	376240.90	1.12	217.3586
u1817	twice around the tree	57201	89396.10	1.56	9.0384
u1817	christofides	57201	66796.26	1.17	163.5775
rl1889	twice around the tree	316536	449335.91	1.42	9.4231
rl1889	christofides	316536	344691.95	1.09	90.8486
d2103	twice around the tree	80450	140961.11	1.75	12.5893
d2103	christofides	80450	84509.81	1.05	23.7074
u2152	twice around the tree	64253	101570.27	1.58	11.7739
u2152	christofides	64253	73917.83	1.15	244.0866
u2319	twice around the tree	234256	405732.66	1.73	15.1885
u2319	christofides	234256	272766.52	1.16	912.3110
pr2392	twice around the tree	378032	526143.63	1.39	18.4906
pr2392	christofides	378032	425979.59	1.13	790.8695
pcb3038	twice around the tree	137694	196718.24	1.43	27.5733
pcb3038	christofides	137694	154218.19	1.12	1763.7831
fl3795	twice around the tree	28772	40756.45	1.42	54.5166
fl3795	christofides	28772	nan	nan	1800.0
fnl4461	twice around the tree	182566	254728.16	1.40	66.6000
fnl4461	christofides	182566	nan	nan	1800.0
rl5915	twice around the tree	565530	858988.73	1.52	102.6747
rl5915	christofides	565530	nan	nan	1800.0
rl5934	twice around the tree	556045	842363.53	1.51	98.5979
rl5934	christofides	556045	nan	nan	1800.0

Table 3: Tabela com dados de execução das instâncias do TSP.