Trabalho Prático 2
Algoritmos para o Problema do Caixeiro Viajante Euclidiano

Indra Matsiendra Cardoso Dias Ribeiro[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

imcdindra@hotmail.com

Abstract. *This article presents details about the implementation of 3 algorithms for solving the Euclidean traveling salesman problem, twice-around-the-tree, Christofides algorithm and branch-and-bound. In this context, experiments were carried out evaluating the performance of each of them in 78 different datasets, taking into account memory spent, execution time and approximation factor.*

Resumo. *Este artigo apresenta detalhes sobre a implementação de 3 algoritmos para a resolução do problema do caixeiro viajante euclidiano, twice-around-the-tree, algoritmo de Christofides e branch-and-bound. Nesse contexto, foram feitos experimentos avaliando o desempenho de cada um deles em 78 datasets diferentes, levando em conta memória gasta, tempo de execução e fator de aproximação.*

1. Introdução

O presente trabalho tem por objetivo comparar o desempenho dos algoritmos twice-around-the-tree, christofides e branch-and-bound com relação a tempo gasto na execução, memória e fator aproximativo em relação ao ótimo. Os algoritmos, disponíveis em `https://github.com/IndraMatsiendra/TP2-algoritmos2.git`, foram implementados em python e cada um deles foi executado em 78 datasets diferentes pertencentes à TSPLIB. Nesse sentido, cabe dizer aqui que este trabalho também é um estudo comparativo sobre cenários onde é mais útil utilizar algoritmos aproximativos ou exatos (branch and bound).

2. Implementação

2.1. Estruturas de dados utilizadas

Os datasets da TSPLIB escolhidos para este trabalho contêm um conjunto de pontos com coordenadas 2D de um espaço euclidiano e, consequentemente, podem ser usados tanto no branch and bound quanto nos algoritmos aproximativos.

Sendo assim, pode-se representar esse espaço por meio de um grafo completamnte conexo, já que sempre é possível ir de um ponto para qualquer outro. Dito isso, fica claro que a melhor opção para grafos grandes desse tipo seria utilizar uma matriz de adjacências ao invés de uma lista de adjacências, pois o acesso aos pesos das arestas seria mais fácil, permitindo uma melhor localidade de referência espacial para o programa como um todo, além de ocupar menos espaço que os ponteiros da lista de adjacências.

Entretanto, para fins de facilitar a implementação, os grafos utlizados nos algoritmos de Christofides e twice-around-the-tree são do pacote Networkx, que são criados por default como listas de adjacências.

Todavia, a classe de grafos da Networks é otimizada para grafos esparsos, onde a maioria das entradas em uma matriz de adjacências seria zero. Levando em conta que nosso conjunto de datasets representam grafos completamente conexos, isso poderia levar a uma sobrecarga associada à estrutura de dados de lista de adjacências. Por isso, o grafo foi representado como matriz de adjacências apenas no algoritmo de branch and bound onde o uso de memória é um fator crítico.

2.2. Twice-around-the-tree

O algoritmo twice-around-the-tree é um algoritmo de aproximação para o problema do Caixeiro Viajante (TSP) que só funciona em instâncias euclidianas. Ele consiste em construir uma árvore geradora mínima(MST) a partir do grafo, transformar a MST em um caminho euleriano duplicando todas as arestas e, por fim, converter o caminho euleriano em um caminho hamiltoniano removendo repetições de vértices, mantendo a ordem da primeira ocorrência de cada vértice.

Esse algoritmo tem fator aproximativo 2 e tem complexidade de tempo dominada pela geração da MST, portanto, tem complexidade O(E log V), onde E é o número de arestas do grafo e V o número de vértices.

2.3. Christofides

O algoritmo de Christofides também é um algoritmo aproximativo para o Problema do Caixeiro Viajante (TSP) que opera somente em instâncias euclidianas. Ele começa construindo uma Árvore Geradora Mínima (MST) no grafo de entrada. Em seguida, calcula o matching perfeito de peso mínimo para o subgrafo induzido pelos vértices de grau ímpar da MST. Depois, ele encontra um Caminho Euleriano no grafo formado pela união da MST com as arestas do matching perfeito. Por fim, esse caminho é transformado em um Caminho Hamiltoniano, removendo repetições de vértices.

Esse algoritmo possui um fator de aproximação de 1.5 e tem complexidade de tempo dominada pela geração do matching de peso mínimo pelo algoritmo de blossom, portanto tem complexidade de tempo O($|V|^3$).

2.4. Branch-and-bound

O algoritmo de branch and bound é exato, e consiste em realizar uma busca exaustiva explorando o espaço de todas soluções possíveis e realizando podas nos ramos com base em um limite inferior, caso o ramo não seja promissor e não valha a pena explorá-lo.Em seu pior caso esse algoritmo exploraria todos os ramos, tendo complexidade de tempo O($|V|!$).

A implementação feita neste trabalho calculou o limite inferior de um nó, somando o custo atual com as duas arestas de menor peso de cada vértice ainda não incluído na solução. Para evitar que a busca das duas arestas de peso mínimo fosse feita muitas vezes piorando o desempenho, esses valores foram salvos num array, aumentando o gasto de memória em O($|V|$).

Além disso, o branch and bound foi implementado realizando a busca em formato depth first, pois a abordagem best-first estoura o limite de memória do computador mesmo para o menor dataset usado nos experimentos.

Por fim, outra otimização utilizada foi executar primeiro os algoritmos de twice-around-the-tree e Christofides e usar a melhor resposta como limite inferior inicial, para assim, realizar mais podas ao longo da execução.

3. Experimentos

Os experimentos realizados consistiram em executar os três algoritmos supramencionados em 78 datasets, limitando o tempo de execução a 30 minutos. Eles foram realizados em um computador com 16GB de RAM, sistema operacional Ubuntu 22.04.

As diferentes execuções são apresentadas na tabela abaixo, que trás o nome do dataset, o número de vétices do grafo, o algoritmo executado, a resposta dada por ele, o valor ótimo de resposta, o fator aproximativo obtido naquela execução (resposta/ótimo) e o tempo gasto em segundos. Nessa tabela, o símbolo ”NA”significa que o algoritmo excedeu o tempo de 30 minutos, foi interrompido e seus dados não estão disponíveis, e o símbolo ”*”significa que o processo foi morto antes mesmo de 30 minutos por estourar o limite de memória disponível no computador.

Vale destacar que os algoritmos de twice-around-the-tree e Christofides, têm fator aproximativo 2 e 1.5, teoricamente, mas, na prática, muitas vezes, eles podem ter um resultado melhor que esse limite. Desse modo, as médias obtidas para eles nos experimentos foram 1.37 e 1.35, respectivamente.

Tabela 1. Resultado dos experimentos

Dataset	Nº vértices	Algoritmo	Resposta	Ótimo	Fator	Tempo (seg.)
eil51	51	twice around the tree	640.90	426	1.50	0.11
eil51	51	christofides	558.87	426	1.31	0.57
eil51	51	branch and bound	NA	426	NA	NA
berlin52	52	twice around the tree	10116.01	7542	1.34	0.01
berlin52	52	christofides	9071.64	7542	1.20	0.07
berlin52	52	branch and bound	NA	7542	NA	NA
st70	70	twice around the tree	873.35	675	1.29	0.02
st70	70	christofides	881.39	675	1.31	0.16
st70	70	branch and bound	NA	675	NA	NA
eil76	76	twice around the tree	707.15	538	1.31	0.01
eil76	76	christofides	686.99	538	1.28	0.24
eil76	76	branch and bound	NA	538	NA	NA
pr76	76	twice around the tree	145338.11	108159	1.34	0.01
pr76	76	christofides	129685.65	108159	1.20	0.10
pr76	76	branch and bound	NA	108159	NA	NA
rat99	99	twice around the tree	1723.23	1211	1.42	0.01
rat99	99	christofides	1577.49	1211	1.30	0.22
rat99	99	branch and bound	*	1211	*	*
kroA100	100	twice around the tree	27211.68	21282	1.28	0.01

Dataset	Nº vértices	Algoritmo	Resposta	Ótimo	Fator	Tempo (seg.)
kroA100	100	christofides	30273.37	21282	1.42	0.29
kroA100	100	branch and bound	*	21282	*	*
kroB100	100	twice around the tree	25881.20	22141	1.17	0.01
kroB100	100	christofides	26947.76	22141	1.22	0.20
kroB100	100	branch and bound	*	22141	*	*
kroC100	100	twice around the tree	27966.54	20749	1.35	0.01
kroC100	100	christofides	27898.60	20749	1.34	0.29
kroC100	100	branch and bound	*	20749	*	*
kroD100	100	twice around the tree	27113.30	21294	1.27	0.01
kroD100	100	christofides	28268.82	21294	1.33	0.28
kroD100	100	branch and bound	*	21294	*	*
kroE100	100	twice around the tree	30507.42	22068	1.38	0.01
kroE100	100	christofides	30206.23	22068	1.37	0.37
kroE100	100	branch and bound	*	22068	*	*
rd100	100	twice around the tree	10791.66	7910	1.36	0.03
rd100	100	christofides	11126.06	7910	1.41	0.49
rd100	100	branch and bound	*	7910	*	*
eil101	101	twice around the tree	830.54	629	1.32	0.01
eil101	101	christofides	826.66	629	1.31	0.44
eil101	101	branch and bound	*	629	*	*
lin105	105	twice around the tree	19498.41	14379	1.36	0.02
lin105	105	christofides	20268.32	14379	1.41	0.28
lin105	105	branch and bound	*	14379	*	*
pr107	107	twice around the tree	54238.04	44303	1.22	0.02
pr107	107	christofides	57503.04	44303	1.30	0.20
pr107	107	branch and bound	*	44303	*	*
pr124	124	twice around the tree	74140.96	59030	1.26	0.02
pr124	124	christofides	79764.69	59030	1.35	0.16
pr124	124	branch and bound	*	59030	*	*
bier127	127	twice around the tree	158637.61	118282	1.34	0.02
bier127	127	christofides	141204.49	118282	1.19	0.78
bier127	127	branch and bound	*	118282	*	*
ch130	130	twice around the tree	8128.76	6110	1.33	0.02
ch130	130	christofides	7851.98	6110	1.29	0.39
ch130	130	branch and bound	*	6110	*	*
pr136	136	twice around the tree	151913.74	96772	1.57	0.04
pr136	136	christofides	127094.11	96772	1.31	0.13
pr136	136	branch and bound	*	96772	*	*
pr144	144	twice around the tree	80596.32	58537	1.38	0.03
pr144	144	christofides	66959.30	58537	1.14	0.16
pr144	144	branch and bound	*	58537	*	*
ch150	150	twice around the tree	8333.47	6528	1.28	0.04
ch150	150	christofides	8431.23	6528	1.29	0.45
ch150	150	branch and bound	*	6528	*	*
kroA150	150	twice around the tree	35122.57	26524	1.32	0.03
kroA150	150	christofides	37660.25	26524	1.42	0.99

Dataset	Nº vértices	Algoritmo	Resposta	Ótimo	Fator	Tempo (seg.)
kroA150	150	branch and bound	*	26524	*	*
kroB150	150	twice around the tree	36154.73	26130	1.38	0.03
kroB150	150	christofides	35377.93	26130	1.35	1.06
kroB150	150	branch and bound	*	26130	*	*
pr152	152	twice around the tree	87998.69	73682	1.19	0.05
pr152	152	christofides	96784.66	73682	1.31	0.15
pr152	152	branch and bound	*	73682	*	*
u159	159	twice around the tree	57787.97	42080	1.37	0.04
u159	159	christofides	55609.56	42080	1.32	0.58
u159	159	branch and bound	*	42080	*	*
rat195	195	twice around the tree	3317.72	2323	1.43	0.06
rat195	195	christofides	2908.38	2323	1.25	1.15
rat195	195	branch and bound	*	2323	*	*
d198	198	twice around the tree	19218.43	15780	1.22	0.07
d198	198	christofides	19809.17	15780	1.26	1.32
d198	198	branch and bound	*	15780	*	*
kroA200	200	twice around the tree	40030.87	29368	1.36	0.06
kroA200	200	christofides	41634.68	29368	1.42	2.11
kroA200	200	branch and bound	*	29368	*	*
kroB200	200	twice around the tree	40710.96	29437	1.38	0.07
kroB200	200	christofides	39029.22	29437	1.33	1.62
kroB200	200	branch and bound	*	29437	*	*
ts225	225	twice around the tree	188009.70	126643	1.48	0.08
ts225	225	christofides	160149.44	126643	1.26	0.34
ts225	225	branch and bound	*	126643	*	*
tsp225	225	twice around the tree	5115.93	3919	1.31	0.09
tsp225	225	christofides	5542.99	3919	1.41	1.53
tsp225	225	branch and bound	*	3919	*	*
pr226	226	twice around the tree	116692.15	80369	1.45	0.08
pr226	226	christofides	120829.39	80369	1.50	1.11
pr226	226	branch and bound	*	80369	*	*
gil262	262	twice around the tree	3358.15	2378	1.41	0.12
gil262	262	christofides	3275.54	2378	1.38	3.50
gil262	262	branch and bound	*	2378	*	*
pr264	264	twice around the tree	66466.84	49135	1.35	0.12
pr264	264	christofides	70563.53	49135	1.44	1.09
pr264	264	branch and bound	*	49135	*	*
a280	280	twice around the tree	3629.72	2579	1.41	0.13
a280	280	christofides	3638.28	2579	1.41	2.77
a280	280	branch and bound	*	2579	*	*
pr299	299	twice around the tree	64646.03	48191	1.34	0.15
pr299	299	christofides	64459.36	48191	1.34	3.55
pr299	299	branch and bound	*	48191	*	*
lin318	318	twice around the tree	58145.44	42029	1.38	0.18
lin318	318	christofides	58811.55	42029	1.40	4.64
lin318	318	branch and bound	*	42029	*	*

Dataset	Nº vértices	Algoritmo	Resposta	Ótimo	Fator	Tempo (seg.)
linhp318	318	twice around the tree	58145.44	41345	1.41	0.20
linhp318	318	christofides	58811.55	41345	1.42	4.74
linhp318	318	branch and bound	*	41345	*	*
rd400	400	twice around the tree	20250.38	15281	1.33	0.29
rd400	400	christofides	20733.22	15281	1.36	14.56
rd400	400	branch and bound	*	15281	*	*
fl417	417	twice around the tree	16297.17	11861	1.37	0.31
fl417	417	christofides	18164.02	11861	1.53	5.52
fl417	417	branch and bound	*	11861	*	*
pr439	439	twice around the tree	144624.16	107217	1.35	0.34
pr439	439	christofides	143720.08	107217	1.34	5.78
pr439	439	branch and bound	*	107217	*	*
pcb442	442	twice around the tree	69364.90	50778	1.37	0.35
pcb442	442	christofides	68437.96	50778	1.35	10.83
pcb442	442	branch and bound	*	50778	*	*
d493	493	twice around the tree	45427.09	35002	1.30	0.49
d493	493	christofides	46698.03	35002	1.33	18.23
d493	493	branch and bound	*	35002	*	*
u574	574	twice around the tree	49083.15	36905	1.33	0.61
u574	574	christofides	49881.75	36905	1.35	29.97
u574	574	branch and bound	*	36905	*	*
rat575	575	twice around the tree	9438.93	6773	1.39	0.61
rat575	575	christofides	9339.49	6773	1.38	34.27
rat575	575	branch and bound	*	6773	*	*
p654	654	twice around the tree	49731.90	34643	1.44	0.83
p654	654	christofides	50576.02	34643	1.46	5.24
p654	654	branch and bound	*	34643	*	*
d657	657	twice around the tree	65730.19	48912	1.34	0.81
d657	657	christofides	64235.18	48912	1.31	44.90
d657	657	branch and bound	*	48912	*	*
u724	724	twice around the tree	57779.67	41910	1.38	1.02
u724	724	christofides	58225.20	41910	1.39	56.19
u724	724	branch and bound	*	41910	*	*
rat783	783	twice around the tree	12054.45	8806	1.37	1.35
rat783	783	christofides	12239.75	8806	1.39	84.84
rat783	783	branch and bound	*	8806	*	*
pr1002	1002	twice around the tree	342244.46	259045	1.32	2.16
pr1002	1002	christofides	351287.36	259045	1.36	128.43
pr1002	1002	branch and bound	*	259045	*	*
u1060	1060	twice around the tree	302914.87	224094	1.35	2.43
u1060	1060	christofides	298423.43	224094	1.33	174.28
u1060	1060	branch and bound	*	224094	*	*
vm1084	1084	twice around the tree	315268.35	239297	1.32	2.58
vm1084	1084	christofides	331618.54	239297	1.39	88.78
vm1084	1084	branch and bound	*	239297	*	*
pcb1173	1173	twice around the tree	80694.89	56892	1.42	3.38

Dataset	Nº vértices	Algoritmo	Resposta	Ótimo	Fator	Tempo (seg.)
pcb1173	1173	christofides	81728.38	56892	1.44	160.33
pcb1173	1173	branch and bound	*	56892	*	*
d1291	1291	twice around the tree	74658.30	50801	1.47	3.64
d1291	1291	christofides	68461.82	50801	1.35	42.71
d1291	1291	branch and bound	*	50801	*	*
rl1304	1304	twice around the tree	347782.57	252948	1.37	3.96
rl1304	1304	christofides	365737.56	252948	1.45	42.65
rl1304	1304	branch and bound	*	252948	*	*
rl1323	1323	twice around the tree	380421.59	270199	1.41	4.01
rl1323	1323	christofides	370646.37	270199	1.37	43.57
rl1323	1323	branch and bound	*	270199	*	*
nrw1379	1379	twice around the tree	79156.27	56638	1.40	4.48
nrw1379	1379	christofides	79121.30	56638	1.40	565.52
nrw1379	1379	branch and bound	*	56638	*	*
fl1400	1400	twice around the tree	27915.52	20127	1.39	4.27
fl1400	1400	christofides	32594.00	20127	1.62	532.19
fl1400	1400	branch and bound	*	20127	*	*
u1432	1432	twice around the tree	214430.78	152970	1.40	4.57
u1432	1432	christofides	207866.92	152970	1.36	169.35
u1432	1432	branch and bound	*	152970	*	*
fl1577	1577	twice around the tree	31223.10	22249	1.40	5.45
fl1577	1577	christofides	30887.09	22249	1.39	107.95
fl1577	1577	branch and bound	*	22249	*	*
d1655	1655	twice around the tree	85681.08	62128	1.38	6.26
d1655	1655	christofides	84585.72	62128	1.36	216.03
d1655	1655	branch and bound	*	62128	*	*
vm1748	1748	twice around the tree	444658.63	336556	1.32	7.25
vm1748	1748	christofides	475614.85	336556	1.41	324.70
vm1748	1748	branch and bound	*	336556	*	*
u1817	1817	twice around the tree	83501.56	57201	1.46	7.77
u1817	1817	christofides	79086.04	57201	1.38	245.94
u1817	1817	branch and bound	*	57201	*	*
rl1889	1889	twice around the tree	449811.49	316536	1.42	9.10
rl1889	1889	christofides	435391.86	316536	1.38	137.02
rl1889	1889	branch and bound	*	316536	*	*
d2103	2103	twice around the tree	124533.87	80450	1.55	10.89
d2103	2103	christofides	93230.69	80450	1.16	32.14
d2103	2103	branch and bound	*	80450	*	*
u2152	2152	twice around the tree	95076.22	64253	1.48	10.86
u2152	2152	christofides	88618.51	64253	1.38	356.55
u2152	2152	branch and bound	*	64253	*	*
u2319	2319	twice around the tree	320532.84	234256	1.37	12.76
u2319	2319	christofides	318199.37	234256	1.36	1354.49
u2319	2319	branch and bound	*	234256	*	*
pr2392	2392	twice around the tree	523132.91	378032	1.38	14.61
pr2392	2392	christofides	527655.03	378032	1.40	1251.62

Dataset	Nº vértices	Algoritmo	Resposta	Ótimo	Fator	Tempo (seg.)
pr2392	2392	branch and bound	*	378032	*	*
pcb3038	3038	twice around the tree	196573.90	137694	1.43	23.66
pcb3038	3038	christofides	NA	137694	NA	NA
pcb3038	3038	branch and bound	*	137694	*	*
fl3795	3795	twice around the tree	39072.83	28772	1.36	36.83
fl3795	3795	christofides	NA	28772	NA	NA
fl3795	3795	branch and bound	*	28772	*	*
fnl4461	4461	twice around the tree	254671.63	182566	1.39	59.45
fnl4461	4461	christofides	NA	182566	NA	NA
fnl4461	4461	branch and bound	*	182566	*	*
rl5915	5915	twice around the tree	840348.36	565530	1.49	109.28
rl5915	5915	christofides	NA	565530	NA	NA
rl5915	5915	branch and bound	*	565530	*	*
rl5934	5934	twice around the tree	827482.34	556045	1.49	501.05
rl5934	5934	christofides	NA	556045	NA	NA
rl5934	5934	branch and bound	*	556045	*	*
rl11849	11849	twice around the tree	*	923368	*	*
rl11849	11849	christofides	*	923368	*	*
rl11849	11849	branch and bound	*	923368	*	*
usa13509	13509	twice around the tree	*	19982889	*	*
usa13509	13509	christofides	*	19982889	*	*
usa13509	13509	branch and bound	*	19982889	*	*
brd14051	14051	twice around the tree	*	469445	*	*
brd14051	14051	christofides	*	469445	*	*
brd14051	14051	branch and bound	*	469445	*	*
d15112	15112	twice around the tree	*	1573152	*	*
d15112	15112	christofides	*	1573152	*	*
d15112	15112	branch and bound	*	1573152	*	*
d18512	18512	twice around the tree	*	645488	*	*
d18512	18512	christofides	*	645488	*	*
d18512	18512	branch and bound	*	645488	*	*

4. Conclusão

Conclui-se que os algoritmos aproximativos podem ser muito vantajosos, apesar de não darem a resposta ótima, tendo em vista o quanto gastam menos recursos do que o branch and bound. Além disso, é possível perceber a diferença de tempo de execução mesmo entre os próprios algoritmos aproximativos, geralmente, quanto menor o fator de aproximação, maior o tempo gasto executando. Isso pode ser verificado pela tabela da seção anterior, em que o algoritmo de Christofides resultou em NA para intâncias maiores que 2392 vértices, enquanto o twice-around-the-tree ainda executava rapidamente. Portanto, o uso de algoritmos exatos só vale a pena para instâncias menores.

5. Referências

KOKMOTOS, Marios. **The Travelling Salesman Problem: An Implementation in Python**. Medium, 2023. Disponível em: https://medium.com/@marioskokmotos2/the-

travelling-salesman-problem-an-implementation-in-python-d2b87e48b9d9. Acesso em: 03 de julho de 2023.

VIMIEIRO, Renato. **Algoritmos Aproximativos**. Apresentação de slides, Disciplina de algoritmos II, Universidade Federal de Minas Gerais, novembro de 2023. Disponível em: https://ufmgbr.sharepoint.com/:b:/r/sites/Algoritmos220232/Material%20de%20Aula/slides/aula13-alg-aproximativos.pdf?csf=1web=1e=zQxrHo

NETWORKX. Documentation. Disponível em: https://networkx.org/documentation/stable/index.html. Acesso em: 03 de julho de 2023.