Algoritmos exato e aproximados para o Problema do Caixeiro Viajante

Lucas Junqueira[1], **Marcos Daniel Souza Netto**[1]

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

lucas.junqueira@dcc.ufmg.br, marcos.netto@dcc.ufmg.br

Abstract. *This work conducts experimentation on approximate and exact solutions for the Traveling Salesman Problem (TSP). To this end, the problem is introduced, followed by a presentation and detailed explanation of the implemented algorithms used to solve it. Finally, a comparative study is conducted, examining the algorithms, problem instances, solution quality (for approximate algorithms), and the time required to reach a solution.*

Resumo. *O presente trabalho realiza experimentação nas soluções aproximadas e exatas para o problema do caixeiro viajante (TSP). Para tanto, apresenta-se o problema, faz-se uma apresentação e detalhamento dos algoritmos implementados para resolvê-lo e, por fim, realiza-se um estudo comparativo entre esses algoritmos, instâncias do problema e qualidade da resposta (no caso dos algoritmos aproximativos), além de ser apresentado o tempo gasto para alcançar cada solução.*

1. Introdução

1.1. Apresentação do problema

O problema do caixeiro viajante é o problema de determinar a menor rota para percorrer uma série de cidades, uma única vez cada uma delas, e retornando à cidade inicial. É um problema NP-difícil da área de otimização combinatória e muito importante tanto para a ciência da computação teórica quanto para áreas práticas como pesquisa operacional. Vale destacar que não são conhecidas soluções exatas polinomiais para o problema. Devido à sua relevância prática, torna-se importante a busca de soluções que, ou acelerem a exploração do espaço de soluções, utilizando o *branch-and-bound*, ou sejam aproximadas, ou seja, que encontram uma resposta com garantia de aproximação da solução ótima mas utilizando algoritmos polinomiais.

1.2. Objetivo

Este trabalho tem como objetivo fazer uma comparação entre o desempenho dos algoritmos *Branch-and-bound*, *Twice-around-the-tree* e *Christofides* na resolução do problema do caixeiro viajante, além de verificar as diferenças entre algoritmos exatos e algoritmos aproximativos, quando executados na prática. Para isso, os três algoritmos foram implementados na linguagem python e testados em 78 instâncias do problema. A implementação completa dos algoritmos pode ser verificada em `https://github.com/souza-marcos/TSP_Exact_and_Approximate`.

2. Implementações

2.1. Branch-and-Bound

O algoritmo Branch-and-Bound gera uma solução exata para o problema do caixeiro viajante e funciona realizando uma busca exaustiva na árvore de soluções de uma determinada instância do problema TSP. Contudo o algoritmo não avalia, necessariamente, todas as soluções possíveis, pois ele, com base em um limite inferior (equivalente à melhor solução encontrada até o momento), realiza podas em ramos da árvore que não são promissores, de modo a otimizar a execução do programa.

Neste trabalho, o limite inferior foi determinado como a soma do custo do caminho encontrado pelo algoritmo até um determinado nó e as arestas de menor peso que estão ligadas aos vértices que ainda não foram visitados. Além disso, foi utilizada uma fila de prioridades para simular o camihamento na árvore de soluções do Branch-and-bound de acordo com a metodologia 'best first search', ou seja, os nós mais promissores da árvore de soluções são visitados primeiro.

2.2. Algoritmos Aproximativos

Os algoritmos aproximativos resolvem um problema de otimização de forma aproximada por meio de algoritmos polinomiais. Esses algoritmos, como já mencionado anteriormente, apresentam uma garantia de qualidade da resposta em relação ao ótimo. Como discutido em [Cormen et al. 2022], o problema do caixeiro viajante, em particular, não possui algoritmo aproximativo de fator constante, salvo se $P = NP$. Dessa forma, vamos tratar de um subproblema do problema original ao restringir os pesos das arestas do grafo para que obedeçam a desigualdade triangular, isto é:

$$\forall x, y, z \in V(G) : d(x, z) \leq d(x, y) + d(y, z)$$

onde $d(v, u)$ é a função de distância que atribui peso à aresta vu.

2.2.1. Twice-Around-The-Tree

Essa abordagem apresenta uma resposta 2-aproximada da solução ótima e seu algoritmo é da seguinte forma:

1. Construção da árvore geradora mínima (MST) do grafo original. Utilizando a biblioteca *networkx*, fazemos chamada ao método de MST baseado no algoritmo de Kruskal.
2. Realizamos um caminhamento pré-ordem nessa árvore, por meio da execução do algoritmo *DFS*, que também é fornecido pela biblioteca citada anteriormente.
3. Transformamos esse caminhamento em um ciclo hamiltoniano, ligando o vértice final ao inicial do caminhamento.

2.2.2. Christofides

Essa abordagem apresenta uma resposta 1.5-aproximada da solução ótima. Note que esse método é derivado do apresentado acima e seu algoritmo é da seguinte forma:

1. Construção da árvore geradora mínima do grafo original(T), utilizando a biblioteca *networkx*.
2. Computamos os vértices que têm grau ímpar em T.
3. A partir do subgrafo induzido no grafo original por esses vértices, computamos um emparelhamento perfeito de peso mínimo (M). Isso pode ser feito utilizando uma chamada para a biblioteca, que executa o algoritmo de Blossom internamente.
4. A partir do grafo da união das arestas de T e M, realizamos um circuito euleriano.
5. Por fim, transformamos esse circuito em um ciclo hamiltoniano.

3. Experimentação

Os três algoritmos implementados no trabalho foram testados em um dataset composto por 78 instâncias compiladas fornecidas pela biblioteca TSPLIB [Heidelber 2025], sendo que, foram utilizadas apenas as instâncias que possuiam como função de custo a distância euclidiana em 2D.

Os testes foram executados com um limite de tempo de 30 minutos e, além do custo obtido por cada algoritmo, foram coletados dados em relação ao tempo de execução e a memória utilizada. Os resultados obtidos estão apresentados na tabela abaixo, onde o símbolo 'NAN' representa que o teste foi abortado por ultrapassar o limite de tempo de 30 minutos e o símbolo '*' representa que o teste foi abortado por ultrapassar o limite de memória disponibilizado pelo computador.

A tabela com os dados estatísticos pode ser encontrada no Anexo I deste documento. É importante notar que os dados sobre o algoritmo *Branch-and-bound* não foi apresentada, uma vez que, para todos os casos de teste, foi alcançado o limite de tempo de 30 minutos.

4. Análise dos Dados

Por causa da limitação do uso do algoritmo *Branch-and-bound*, que foi apontada anteriormente, será feita uma análise de dados sobre os algoritmos aproximativos. Assim, mostraremos como os algoritmos reagem a diferentes instâncias, em que o foco principal será a variação da quantidade de vértices.

A figura 1 apresenta como o fator de qualidade muda em diferentes instâncias do problema. É fácil perceber que em todas as instâncias testadas, o limite do fator de aproximação não é alcançado e, muitas vezes, os algoritmos apresentam respostas muito boas.

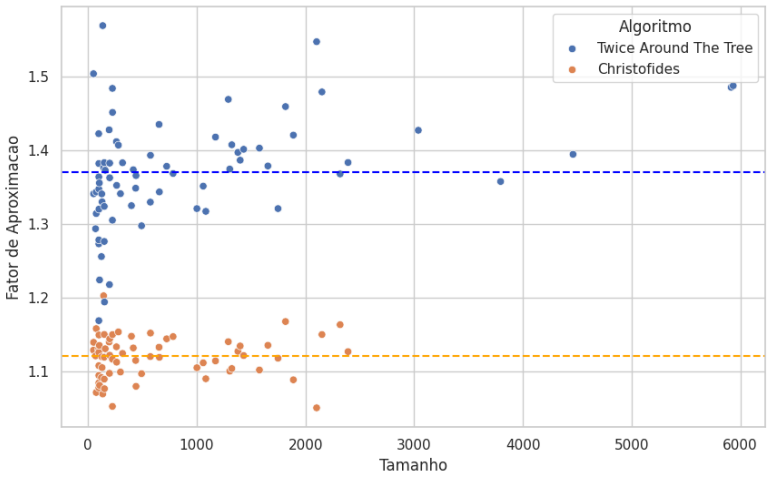

Figura 1. Fator de aproximação vs. Tamanho

Já na figura 2, conseguimos perceber como os tempos de execução dos algoritmos se comportam com relação a variação do tamanho da instância, isto é, o número de vértices do grafo. A diferença de complexidade dos dois algoritmos pode ser notada no grafo, uma vez que o Twice-around-the-tree apresenta complexidade $O(n^2)$ no número de vértices e o Christofides apresenta complexidade $O(n^3)$. Ademais, ao passo que o primeiro algoritmo consegue resolver todas as instâncias (até aproximadamente 6000 vértices) no limite de tempo de 30 minutos, o segundo já não consegue a partir de 3000 vértices.

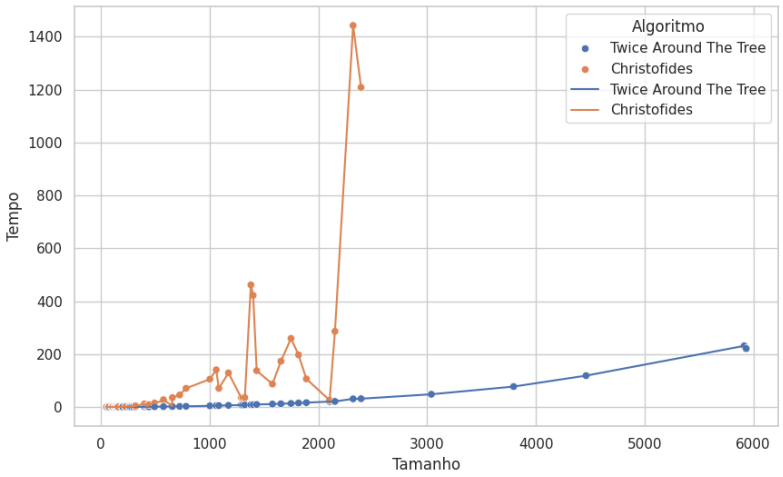

Figura 2. Tempo gasto vs. Tamanho

5. Conclusões

Com a realização desde trabalho, foi possível perceber a real diferença prática entre algoritmos exatos e algoritmos aproximativos, pois, enquanto o algoritmo exato Branch-and-bound não foi capaz de alcançar nenhuma solução em menos de 30 minutos (tempo limite

de execução), os algoritmos aproximados conseguiram encontrar uma solução aproximada para quase todas as instâncias de teste dentro deste intervalo de tempo. Além disso, é possível perceber a diferença entre algoritmos com diferentes fatores de aproximação, já que o algoritmo Christofides (fator de aproximação 1.5) é notavelmente mais lento que o algoritmo Twice-around-the-tree (fator de aproximação 2), o que revela a necessidade de ter de se escolher um balenceamento entre exatidão e velocidade computacional para a resolução de problemas NP-Difíceis.

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 4 edition.

Heidelber, R.-K.-U. (2025). Tsplib.

A. Anexo I

Tabela 1: Resultado dos testes.

Instância	**Nº vértices**	**Algoritmo**	**Custo Obtido**	**Custo Ótimo**	**Memória Pico (MB.)**	**Tempo (seg.)**
eil51	51	Christofides	480.88	426	0.05	0.33
	51	TAT	640.90	426	0.01	0.17
berlin52	52	TAT	10116.01	7542	0.01	0.06
	52	Christofides	8594.03	7542	0.03	0.26
st70	70	TAT	873.35	675	0.01	0.17
	70	Christofides	756.71	675	0.08	0.51
eil76	76	Christofides	623.06	538	0.10	0.66
	76	TAT	707.15	538	0.02	0.17
pr76	76	TAT	145338.11	108159	0.02	0.17
	76	Christofides	115878.39	108159	0.05	0.41
rat99	99	Christofides	1367.43	1211	0.11	0.72
	99	TAT	1723.23	1211	0.03	0.39
kroD100	100	TAT	27113.30	21294	0.03	0.40
	100	Christofides	23587.75	21294	0.14	0.78
kroE100	100	TAT	30507.42	22068	0.03	0.40
	100	Christofides	23782.61	22068	0.23	1.06
rd100	100	TAT	10791.66	7910	0.03	0.40
	100	Christofides	8905.49	7910	0.22	1.11
kroC100	100	TAT	27966.54	20749	0.03	0.40
	100	Christofides	23469.13	20749	0.14	0.92
kroA100	100	Christofides	23292.98	21282	0.18	0.94
	100	TAT	27211.68	21282	0.06	0.54
kroB100	100	Christofides	24009.19	22141	0.10	0.68
	100	TAT	25881.20	22141	0.03	0.40
eil101	101	TAT	830.54	629	0.03	0.41
	101	Christofides	723.00	629	0.19	1.22

lin105	105	TAT	19498.41	14379	0.03	0.45
	105	Christofides	16323.68	14379	0.24	0.75
pr107	107	TAT	54238.04	44303	0.04	0.62
	107	Christofides	47894.18	44303	0.11	0.98
pr124	124	TAT	74140.96	59030	0.05	0.68
	124	Christofides	64437.70	59030	0.10	0.68
bier127	127	TAT	158637.61	118282	0.05	0.72
	127	Christofides	132456.25	118282	0.36	1.78
ch130	130	Christofides	6752.28	6110	0.34	1.03
	130	TAT	8128.76	6110	0.05	0.76
pr136	136	Christofides	103478.24	96772	0.10	0.84
	136	TAT	151913.74	96772	0.06	0.99
pr144	144	Christofides	70405.28	58537	0.10	0.96
	144	TAT	80596.32	58537	0.06	0.96
kroB150	150	Christofides	30054.39	26130	0.52	1.85
	150	TAT	36154.73	26130	0.07	1.05
kroA150	150	TAT	35122.57	26524	0.07	1.20
	150	Christofides	29688.32	26524	0.47	1.85
ch150	150	TAT	8333.47	6528	0.07	1.05
	150	Christofides	7112.04	6528	0.27	1.10
pr152	152	Christofides	79314.35	73682	0.11	1.08
	152	TAT	87998.69	73682	0.12	1.08
u159	159	TAT	57787.97	42080	0.08	1.20
	159	Christofides	47581.72	42080	0.29	1.68
rat195	195	Christofides	2648.10	2323	0.69	2.06
	195	TAT	3317.72	2323	0.16	1.87
d198	198	Christofides	17316.37	15780	0.67	2.31
	198	TAT	19218.43	15780	0.13	1.95
kroA200	200	TAT	40030.87	29368	0.12	2.13
	200	Christofides	33602.86	29368	1.10	3.22
kroB200	200	Christofides	33026.94	29437	1.03	2.54
	200	TAT	40710.96	29437	0.20	1.99
ts225	225	Christofides	133282.17	126643	0.29	2.54
	225	TAT	188009.70	126643	0.16	2.54
tsp225	225	Christofides	4376.00	3919	1.28	2.54
	225	TAT	5115.93	3919	0.16	2.54
pr226	226	Christofides	92425.75	80369	0.98	3.06
	226	TAT	116692.15	80369	0.16	2.56
gil262	262	Christofides	2695.15	2378	2.92	4.60
	262	TAT	3358.15	2378	0.22	3.61
pr264	264	TAT	66466.84	49135	0.29	3.55
	264	Christofides	54663.60	49135	0.72	3.55
a280	280	Christofides	2975.11	2579	2.13	4.52
	280	TAT	3629.72	2579	0.29	4.16
pr299	299	TAT	64646.03	48191	0.33	4.73

	299	Christofides	52968.22	48191	2.23	4.95
lin318	318	TAT	58145.44	42029	0.37	5.35
	318	Christofides	47259.76	42029	3.91	6.06
rd400	400	Christofides	17538.63	15281	11.79	11.18
	400	TAT	20250.38	15281	0.63	8.47
fl417	417	Christofides	13424.73	11861	4.57	8.95
	417	TAT	16297.17	11861	0.67	8.95
pr439	439	Christofides	119532.04	107217	4.88	9.98
	439	TAT	144624.16	107217	0.76	10.13
pcb442	442	Christofides	54823.99	50778	9.13	12.06
	442	TAT	69364.90	50778	0.78	10.25
d493	493	TAT	45427.09	35002	1.02	12.76
	493	Christofides	38393.64	35002	14.92	13.05
u574	574	Christofides	41337.61	36905	24.00	17.43
	574	TAT	49083.15	36905	1.35	17.20
rat575	575	Christofides	7802.75	6773	27.48	19.47
	575	TAT	9438.93	6773	1.43	17.25
p654	654	TAT	49731.90	34643	1.80	22.30
	654	Christofides	39244.71	34643	4.13	22.30
d657	657	Christofides	54738.30	48912	35.48	22.49
	657	TAT	65730.19	48912	1.84	22.49
u724	724	Christofides	47954.68	41910	45.39	27.46
	724	TAT	57779.67	41910	2.21	27.46
rat783	783	TAT	12054.45	8806	2.64	32.02
	783	Christofides	10102.87	8806	70.57	40.21
pr1002	1002	Christofides	286233.10	259045	105.35	61.19
	1002	TAT	342244.46	259045	4.24	52.51
u1060	1060	TAT	302914.87	224094	5.17	58.64
	1060	Christofides	249108.58	224094	140.62	59.69
vm1084	1084	Christofides	260833.06	239297	70.33	61.69
	1084	TAT	315268.35	239297	5.23	61.70
pcb1173	1173	Christofides	63392.44	56892	128.81	72.00
	1173	TAT	80694.89	56892	6.22	72.00
d1291	1291	Christofides	57930.37	50801	34.34	86.71
	1291	TAT	74658.30	50801	7.43	86.71
rl1304	1304	TAT	347782.57	252948	7.48	89.17
	1304	Christofides	278319.19	252948	34.32	89.17
rl1323	1323	TAT	380421.59	270199	8.04	91.56
	1323	Christofides	298261.72	270199	35.25	91.56
nrw1379	1379	TAT	79156.27	56638	8.35	99.84
	1379	Christofides	63845.07	56638	461.81	111.24
fl1400	1400	Christofides	22834.35	20127	422.17	163.04
	1400	TAT	27915.52	20127	8.79	102.65
u1432	1432	Christofides	171558.08	152970	137.44	106.98
	1432	TAT	214430.78	152970	9.54	106.98
fl1577	1577	Christofides	24515.85	22249	86.76	130.00

	1577	TAT	31223.10	22249	10.98	130.00
d1655	1655	Christofides	70542.34	62128	172.92	143.64
	1655	TAT	85681.08	62128	12.18	143.64
vm1748	1748	TAT	444658.63	336556	13.37	160.34
	1748	Christofides	376240.90	336556	259.19	160.34
u1817	1817	Christofides	66796.26	57201	197.23	172.15
	1817	TAT	83501.56	57201	15.05	172.15
rl1889	1889	TAT	449811.49	316536	16.39	186.66
	1889	Christofides	344565.85	316536	106.26	186.66
d2103	2103	TAT	124533.87	80450	20.43	231.73
	2103	Christofides	84509.81	80450	25.55	231.73
u2152	2152	TAT	95076.22	64253	20.97	241.74
	2152	Christofides	73895.99	64253	286.18	241.74
u2319	2319	TAT	320532.84	234256	30.75	280.13
	2319	Christofides	272542.91	234256	1442.82	494.20
pr2392	2392	TAT	523132.91	378032	31.28	299.23
	2392	Christofides	425979.59	378032	1208.62	299.22
pcb3038	3038	TAT	196573.90	137694	48.08	482.63
	3038	Christofides	NaN	137694	NaN	NaN
fl3795	3795	Christofides	NaN	28772	NaN	NaN
	3795	TAT	39072.83	28772	77.15	754.33
fnl4461	4461	TAT	254671.63	182566	118.53	1044.83
	4461	Christofides	NaN	182566	NaN	NaN
rl5915	5915	TAT	840348.36	565530	231.04	1823.01
	5915	Christofides	NaN	565530	NaN	NaN
rl5934	5934	TAT	827482.34	556045	220.81	1833.83
	5934	Christofides	NaN	556045	NaN	NaN
rl11849	11849	TAT	*	923368	*	*
	11849	Christofides	*	923368	*	*
usa13509	13509	TAT	*	19982889	*	*
	13509	Christofides	*	19982889	*	*
brd14051	14051	TAT	*	469445	*	*
	14051	Christofides	*	469445	*	*
d15112	15112	TAT	*	1573152	*	*
	15112	Christofides	*	1573152	*	*
d18512	18512	TAT	*	645488	*	*
	18512	Christofides	*	645488	*	*