Relatório Parcial Iniciação Científica (USP)

Título: O Problema do Caixeiro Viajante: Modelos e Aplicações
Bolsista: Gabriel Sanches da Silva - N.USP: 11884693
Orientadora: Franklina M. B. Toledo
Período do Relatório: 09/2022 a 08/2023

1 Introdução

O Problema do Caixeiro Viajante (PCV) (em inglês, *Travelling Salesman Problem (TSP)*), devido a sua relevância prática e ao desafio científico que representa, é um problema muito estudado até os dias de hoje. Ele tem como objetivo encontrar o melhor caminho para que um caixeiro viajante visite um conjunto de cidades, passando em cada uma delas uma única vez, e retornando a cidade inicial. Este problema pode ser representado pelo modelo a seguir.

$$\text{minimizar} \sum_{i=0}^{n} \sum_{j=0, j\neq i}^{n} c_{ij} x_{ij} \tag{1}$$

$$\text{sujeito a} \sum_{i=0, i\neq j}^{n} x_{ij} = 1 \qquad j = 0, \ldots, n \tag{2}$$

$$\sum_{j=0, j\neq i}^{n} x_{ji} = 1 \qquad i = 0, \ldots, n \tag{3}$$

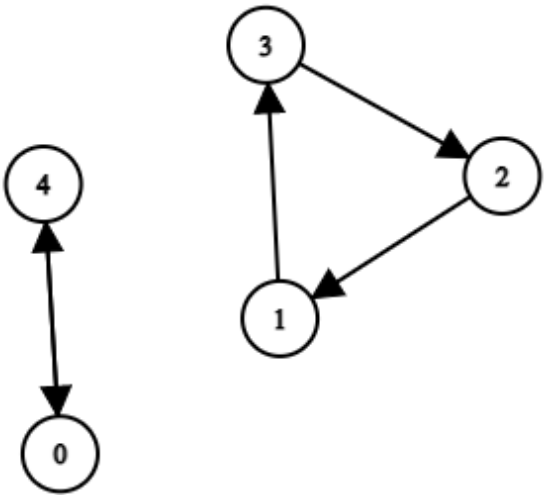
$$x_{ij} \in \{0, 1\} \qquad i, j = 0, \ldots, n \tag{4}$$

Em (1), definimos a função objetivo do problema, que, neste caso, é dada pela soma de cada caminho percorrido multiplicado pelo seu respectivo custo, ou seja, cada caminho possui um custo, ou distância. Queremos minimizar sua soma. As restrições (2) e (3) são responsáveis por garantir que uma cidade possua apenas uma via de entrada e uma via de saída, para evitar que voltemos a mesma cidade mais de uma vez, e para garantir que todas as cidades sejam visitadas. Por fim, (4) definem o domínio das variáveis, e como o problema trata de caminho utilizados, os únicos valores possíveis são 0 (para um caminho não utilizado) e 1 (para um caminho utilizado).

Contudo, nesta modelagem, nada impede que a melhor solução encontrada não tenha subciclos. Isto é, a solução não precisa ser um ciclo contendo todas as cidades, poderia haver subciclos que não se interligam, como ilustra a Figura 1. Este tipo de situação

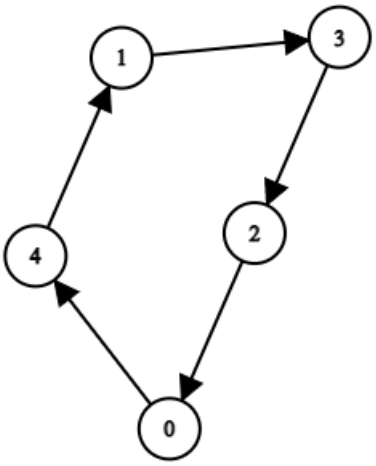
é inviável na prática, pois não há maneira de sair de uma cidade e ir para outra sem percorrer o caminho entre elas.

Figura 1: Exemplo de solução com subciclos (infactível).



Para tanto, diversos modelos e métodos de resolução foram desenvolvidos para impedir os subciclos e obter soluções como a ilustrada na Figura 2. Nesta Iniciação Científica, foram utilizados três modelos matemáticos como objeto de estudo para compreender o problema do caixeiro viajante e diferentes abordagens usadas na literatura para resolvê-lo.

Figura 2: Exemplo de solução sem subciclos (factível).



2 Abordagem utilizada

Na literatura, diversos modelos matemáticos foram propostos para tratar o problema do caixeiro viajante (PCV). Dentre eles, três foram escolhidos para serem estudados neste projeto: i) o modelo de Dantzig et al. [1954] (DFJ); ii) o modelo de Miller et al. [1960] (MTZ); e iii) o modelo de Desrochers and Laporte [1991] (DL). Cada um adiciona novas restrições ao modelo (1) – (4) para evitar a criação de subciclos, visando acelerar a busca por uma solução ótima. Estes modelos são descritos a seguir.

2.1 Modelo de Dantzig-Fulkerson-Johnson

O primeiro modelo estudado foi o modelo DFJ, proposto por Dantzig, Fulkerson e Johnson [1954]. Esta formulação consiste em proibir os subciclos como descrito na restrição (5).

$$
\begin{aligned}
\text{(DFJ) minimizar} & \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij}\ x_{ij} \\
\text{sujeito a } & (2) - (3) \\
& \sum_{i,j \in S} x_{ij} \leq |S| - 1 \qquad S \subseteq \{2, \ldots, n\},\ 2 \leq |S| \leq n-1, \qquad (5) \\
& x_{ij} \in \{0, 1\}, \qquad i, j = 0, \ldots, n.
\end{aligned}
$$

Como existe um número exponencial de restrições (5), uma estratégia utilizada para resolver uma instância utilizando o modelo DFJ é, inicialmente, resolver o problema sem estas restrições de subciclo. Em seguida, à medida que uma solução ótima com subciclos é encontrada, as restrições que os proíbem são adicionadas ao modelo, e o problema é novamente resolvido. Em outras palavras, após uma solução ter sido obtida, verifica-se a presença de subciclos e, se eles forem encontrados, novas restrições são adicionadas ao modelo para que aquele subciclo não se repita. Feito isso, uma nova solução é obtida, e esse procedimento se repete até que uma solução ótima sem subciclos seja encontrada.

Para verificar a existência de subciclos, é utilizada a cardinalidade do conjunto de nós S. Esse conjunto é constituído pelos nós de um subciclo encontrado na solução, e, através de sua cardinalidade, ou seja, o número de nós que o compõe, podemos restringi-lo. O procedimento descrito é uma forma simples de resolver o problema utilizando o modelo de DFJ. Uma estratégia mais eficiente é resolvê-lo utilizando *callbacks*, que permitem controlar a necessidade de inclusão de restrições de subciclo ao longo da resolução do problema. Essa estratégia é resumida a seguir.

2.1.1 Estratégia usando *Callbacks*

Para aprimorar a resolução do modelo DFJ, foi utilizada a *Application Programming Interface* (Interface de Programação de Aplicação), ou API para abreviar, do software comercial Gurobi. Essa interface dispõe de diversas funções e configurações para o modelo que melhoram a resolução do problema, uma dessas funções é chamada de *callback*.

De uma forma geral, uma *callback* é uma função chamada durante a execução de uma outra função. Para o nosso estudo, como proposto na literatura, desenvolvemos uma *callback* que é chamada durante a busca de uma solução, mais especificamente quando uma solução factível é encontrada. Quando chamada, a *callback* adicionará as restrições de subciclos encontradas na solução factível, ao invés de serem adicionadas apenas após uma solução ótima ser encontrada. Isso garante uma enorme melhoria na resolução do problema estudado.

Em resumo, utilizando a API do Gurobi e o conceito de *callback*, o modelo apresentou um aumento drástico na velocidade de resolução.

2.1.2 Adição de restrições de subciclos a priori

Ao longo do estudo do modelo base, como proposto na literatura, também utilizamos como estratégia de resolução adicionar dois conjuntos de restrições a priori visando acelerar a busca por uma solução ótima. Foram adicionadas ao modelo as restrições que proíbem subciclos com duas e com três cidades.

2.2 Modelo de Miller-Tucker-Zemlin

O segundo modelo estudado utiliza uma estratégia diferente para proibir subciclos. Na modelagem de Miller, Tucker e Zemlin [1960], as restrições que proíbem subciclos são dadas por (6) e (7).

$$\text{(MTZ) minimizar} \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij}\, x_{ij}$$

$$\text{sujeito a } (2) - (3)$$

$$u_i - u_j + (n-1)x_{ij} \leq n-2 \qquad i, j = 0, \ldots, n \qquad (6)$$

$$1 \leq u_i \leq n-1 \qquad i = 2, \ldots, n \qquad (7)$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 0, \ldots, n$$

A ideia das restrições (6) e (7) é associar pesos aos nós de forma que, se um caminho indica que o caixeiro viajante sai do nó i e vai para o nó j, então o valor do nó j (u_j)

é maior que o valor do nó i (u_i). Desta forma, não é possível ter um subciclo. De uma maneira simplificada, é como uma contagem iniciando-se do zero: conta-se $0, 1, 2, ..., n$, então não poderia haver duas sequência como $0, 1, 2, 0$ e $3, 4, 5, 3$; o modelo obriga que haja uma sequência de 0 a 5.

Este modelo tem um número polinomial de restrições, logo é possível escrevê-lo mesmo para instâncias com um número grande de nós.

2.3 Modelo de Desrochers-Laporte

O terceiro modelo estudado foi proposto por Desrochers e Laporte [1991]. Os autores propuseram uma melhoria do modelo de Miller et al. [1960]. O modelo Desrochers-Laporte (DL) é descrito a seguir.

$$\text{(DL) minimizar} \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij}\ x_{ij}$$

$$\text{sujeito a } (2) - (3)$$

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2 \qquad i, j = 0, \ldots, n \qquad (8)$$

$$1 + (n-3)x_{i1} + \sum_{j=2}^{n} x_{ji} \leq u_i \qquad i = 2, \ldots, n \qquad (9)$$

$$u_i \leq n - 1 - (n-3)x_{1i} - \sum_{j=2}^{n} x_{ij} \qquad i = 2, \ldots, n \qquad (10)$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 0, \ldots, n$$

Foi comprovado que as restrições (8), (9) e (10) são equivalentes às restrições propostas no modelo MTZ, porém tornam o modelo mais forte.

3 Experimentos Computacionais

No primeiro relatório, apresentamos testes computacionais para os três modelos descritos, sendo que o modelo de DFJ foi resolvido sem utilizar *callback*. Como pudemos observar, o modelo DFJ e o MTZ apresentaram um desempenho inferior ao modelo DL. Este relatório está disponível `https://github.com/TurnipPudding/Instancias_TSP/tree/main`.

Nesta segunda etapa da iniciação científica, a implementação do modelo de DFJ foi aprimorada utilizando *callback*. Novos testes computacionais foram realizados para avaliar seu desempenho em comparação ao modelo DL.

Para avaliar os dois modelos foram utilizadas 20 instâncias geradas a partir da instância uy734 que contém 734 nós. Esta instância está disponível em Cook [2023]. Foram geradas cinco instâncias com 50, 100, 200 e 300 nós escolhidos aleatoriamente a partir da instância uy734. Estas instâncias estão disponíveis em `https://github.com/TurnipPudding/Instancias_TSP/tree/main`.

Os experimentos computacionais foram realizados em uma máquina com as seguintes características: Processador: Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz 3.20 GHz RAM Instalada: 16,0 GB, Sistema operacional Windows 10 de 64 bits, processador baseado em x64. Os testes computacionais foram realizados em três etapas. Primeiramente, resolvemos as instâncias geradas utilizando o modelo DFJ descrito em linguagem Python com a com a API do Gurobi v. 9.5.2. A estratégia de resolução utilizou *callback* sem e com a adição a priori das restrições de subciclos com duas e três cidades. Na segunda etapa, comparamos os melhores resultados obtidos utilizando o modelo DFJ com os resultados obtidos utilizando o modelo DL. Finalmente, na terceira etapa, resolvemos as instâncias utilizando o software não comercial de otimização CBC, usando o modelo DL por ser mais fácil de implementar.

As tabelas que reportam os testes computacionais seguem uma mesma estrutura. A primeira coluna apresenta as instâncias, seguida da coluna com o número de cidades de cada instância. As duas colunas seguintes são separadas em outras quatro colunas menores, sendo elas, respectivamente, o Limitante Inferior (LI), a melhor solução obtida (Sol), o tempo de execução (T) dado em segundos (s) respeitando o limite de 1.800 segundos, e o desvio percentual (GAP) entre a melhor solução encontrada e o limitante inferior dado em porcentagem (%). O GAP foi calculado da seguinte forma: GAP $= 100\frac{Sol-LI}{Sol}$.

3.1 Análise do Modelo DFJ

Na Tabela 1, são apresentados os resultados obtidos considerando o modelo sem a adição a priori das restrições de subciclos com duas e três cidades (s/ RSC23) e com estas restrições (c/ RSC23) para instâncias com até 100 cidades. Como podemos observar, a adição a priori das restrições de subciclo não é vantajosa, pois o tempo de resolução aumenta significativamente.

Tabela 1: Resultados obtidos considerando o modelo DFJ.

Instância	Cidades	s/ RSC23				c/ RSC23			
		LI	**Sol**	**T (s)**	**GAP**	**LI**	**Sol**	**T (s)**	**GAP**
1	50	25730	25730	0,9	0,0	25730	25730	3,4	0,0
2		23743	23743	0,9	0,0	23743	23743	1,4	0,0
3		23440	23440	0,7	0,0	23440	23440	0,9	0,0
4		26238	26238	0,6	0,0	26238	26238	5,3	0,0
5		21933	21933	0.7	0,0	21933	21933	1,3	0,0
6	100	31442	31442	4,1	0,0	31442	31442	29,5	0,0
7		30665	30665	4,7	0,0	30665	30665	43,7	0,0
8		30743	30743	6,3	0,0	30473	30473	125,6	0,0
9		32009	32009	6,2	0,0	32009	32009	60,7	0,0
10		31058	31058	7.2	0,0	31058	31058	62,8	0,0
11	200	41713	41713	100,5	0,0	41710	41713	689,9	<0,1
12		43359	43359	294,0	0,0	43357	43359	866,5	<0,1
13		43607	43607	93,6	0,0	43607	43607	1054,9	0,0
14		42073	42073	115,7	0,0	42073	42073	1383,5	0,0
15		41981	41981	101,6	0,0	41981	41981	924,5	0,0
Gap Médio					0,0				<0,1

3.2 Análise comparativa entre o DFJ e DL

Na Tabela 2, são reportados os resultados obtidos considerando os modelos DFJ e DL utilizando a API do Gurobi para as 20 instâncias geradas. Os resultados mostram que o modelo DFJ s/ RSC23 é o mais indicado para resolver as instâncias apresentadas com até 300 cidades. Para 17 das 20 instâncias, foi possível obter uma solução comprovadamente ótima, e duas delas foram afetadas por um erro de precisão, enquanto utilizando o modelo DL foi possível obter apenas uma solução comprovadamente ótima para apenas 9 instâncias. Em particular, a solução obtida para a instância 15 é ótima, mas o limitante inferior não prova a sua otimalidade.

Tabela 2: Resultados obtidos considerando os modelo DFJ e DL.

Instâncias	Cidades	DFJ s/RSC23				DL			
		LI	**Sol**	**T (s)**	**GAP**	**LI**	**Sol**	**T (s)**	**GAP**
1	50	25730	25730	0,9	0,0	25730	25730	6,1	0,0
2		23743	23743	0,9	0,0	23473	23473	15,9	0,0
3		23440	23440	0,7	0,0	23440	23440	118,1	0,0
4		26238	26238	0,6	0,0	26238	26238	5,9	0,0
5		21933	21933	0,7	0,0	21933	21933	5,7	0,0
6	100	31442	31442	4,1	0,0	31442	31442	463,1	0,0
7		30665	30665	4,7	0,0	30665	30665	108,9	0,0
8		30743	30743	6,3	0,0	30027	30652	1800,0	2,0
9		32009	32009	6,2	0,0	32009	32009	501,5	0,0
10		31058	31058	7,2	0,0	31055	31058	718,8	<0,1
11	200	41713	41713	100,5	0,0	40951	47209	1800,0	13,3
12		43359	43359	294,0	0,0	42528	45633	1800,0	6,8
13		43607	43607	93,6	0,0	42360	45839	1800,0	7,6
14		42073	42073	115,7	0,0	40318	44067	1800,0	8,5
15		41981	41981	101,6	0,0	41479	41981	1800,0	1,2
16	300	50067	50070	1255,5	<0,1	49192	57096	1800,0	13,8
17		51206	51206	571,8	0,0	49957	63203	1800,0	21,0
18		53700	53700	991,8	0,0	52720	63128	1800,0	16,5
19		51819	51987	1800,0	0,3	50635	56984	1800,0	11,1
20		50716	50721	1392,8	<0,1	49378	55644	1800,0	11,3
Gap Médio					<0,1				5,7

3.3 Análise comparativa entre o Gurobi e o CBC

Para estabelecer uma comparação com o Gurobi, buscamos resolver as instâncias utilizando o software não comercial de otimização CBC. Por questão de facilidade e tempo, o modelo DL foi descrito utilizando linguagem Python e a linguagem de modelagem PuLP para esta comparação. Neste período da iniciação científica, não foi possível desenvolver o modelo DFJ utilizando *callback* juntamente à PuLP. Como este modelo já havia se mostrado ineficiente ao utilizar a estratégia simples de eliminação de subciclos, seus resultados utilizando CBC não foram avaliados. Vale destacar que a implementação foi

feita utilizando o PuLP e não a API do Gurobi, portanto, as instâncias foram resolvidas novamente.

Na Tabela 3, são reportados os resultados obtidos. Os dados reportados são os mesmos das tabelas anteriores, sendo que as colunas 3 à 6 apresentam os resultados utilizando o CBC e as colunas 7 à 10 os resultados obtidos utilizando o Gurobi.

Tabela 3: Resultados obtidos considerando o modelo DL.

Instância	Cidades	DL CBC				DL GRB			
		LI	**Sol**	**T (s)**	**GAP**	**LI**	**Sol**	**T (s)**	**GAP**
1	50	25730	25730	333,2	0,0	25730	25730	5,5	0,0
2		22782	26993	1800,0	15,6	23743	23743	16,3	0,0
3		21898	23583	1800,0	7,2	23440	23440	12,0	0,0
4		26238	26238	724,3	0,0	26238	26238	7,3	0,0
5		21933	21933	698,1	0,0	21933	21933	4,0	0,0
6	100	30329	39966	1800,0	24,1	31442	31442	242,6	0,0
7		29918	36245	1800,0	17,5	30665	30665	60,0	0,0
8		29159	35201	1800,0	17,2	30067	30652	1800,0	1,9
9		31026	44087	1800,0	29,6	32009	32009	375,5	0,0
10		30356	32384	1800,0	6,3	31058	31058	227,6	0,0
Gap Médio					11,8				0,2

Como observamos, usando o CBC foi possível encontrar soluções factíveis para instâncias com até 100 cidades. No entanto, Não foi possível encontrar nenhuma solução para as instâncias com mais cidades. Utilizando o Gurobi, foram obtidas soluções factíveis para todas as instâncias, mas, a partir das instâncias com 200 cidades, não foi possível garantir a otimalidade das soluções. Podemos concluir que o CBC é competitivo para instâncias com até 50 cidades.

4 Conclusões e Pesquisas Futuras

De um ponto de vista geral, o modelo DFJ com a API do Gurobi retorna os melhores resultados, seguido do DL utilizando PuLP e Gurobi. O solver não-comercial CBC apresentou os piores resultados, contudo, para instâncias com até 50 cidades, mostrou-se capaz de resolver o problema dentro de um intervalo de 1800 segundos.

Como pesquisas futuras, o uso de *callback* com o PuLP é uma área a ser explorada

para buscar melhores resultados utilizando o modelo DFJ sem o auxílio do solver comercial. Em outras palavras, segundo a literatura, a resolução de problemas utilizando o DFJ de forma bem estruturado deveria obter melhores resultados que o modelo DL, mesmo ao utilizar um solver não-comercial como o CBC.

Referências

W. Cook. Tour of 50 USA landmarks. www.math.uwaterloo.ca/tsp/usa50/, 2023. Acesso em 09/08/2023.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

M. Desrochers and G. Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, 1960.