# 1 Problema do Caixeiro Viajante Tridimensional com restrição de Ordem[1]

**Autores**: Pedro Hokama e Flávio K. Miyazawa

Consideramos um problema que combina os problemas do Caixeiro Viajante (TSP) e do Carregamento em Contêiner, que chamamos por Problema do Caixeiro Viajante Tridimensional (3L-TSP). Neste problema, um veículo deve partir carregado de uma origem e entregar caixas em pontos pré-definidos para seus clientes. Cada cliente tem um conjunto de caixas que deve receber e o objetivo é minimizar o custo de deslocamento do veículo. As caixas devem ser retiradas a partir do fundo do contêiner do veículo e a remoção das caixas de um cliente não podem ser obstruidas pelas caixas a serem descarregadas posteriormente. Assim, diferentes rotas exigem a geração de diferentes configurações de empacotamento.

Para definir este problema, usaremos o espaço $\mathbb{R}^3$ com coordenadas $xyz$. Denotamos uma caixa $b_i$ como uma tripla $b_i = (x_i, y_i, z_i)$. O mesmo pode ser feito para o contêiner $B = (W, H, D)$. Primeiro, definimos o empacotamento dentro de um contêiner. Um empacotamento de uma lista de caixas $L = (b_1, \ldots, b_n)$ no contêiner $B = (W, H, D)$ é dado por uma função $\mathcal{P} : L \to [0, W) \times [0, H) \times [0, D)$, e nenhum item passa dos limites do contêiner e que dois itens não podem se sobrepor.

Para representar empacotamentos que respeitam a ordem de entrega das caixas em uma certa rota, estendemos a definição de empacotamento tridimensional de [7] para considerar também restrições de ordem: Dado uma sequência de listas $\mathcal{L} = (L_1, L_2, \ldots, L_k)$ dizemos que um empacotamento $\mathcal{P}$ de $L = L_1 \cup \ldots \cup L_k$ respeita a sequência de $\mathcal{L}$ na direção $z$ se para toda caixa $b \in L_i$, temos que

$$\mathcal{R}^z(b) \cap \mathcal{R}(c) = \emptyset \quad \text{para toda caixa} \quad c \in L_j \quad \text{e} \quad j > i,$$

onde $\mathcal{R}^z(b) = [\mathcal{P}^x(b_i), \mathcal{P}^x(b_i) + x_i) \times [\mathcal{P}^y(b_i), \mathcal{P}^y(b_i) + y_i) \times [\mathcal{P}^z(b_i), D)$.

Isto é, se $b \in L_i$, não há nenhuma caixa $c$ na frente de $b$ que pertence a um conjunto $L_j$ que será descarregado depois.

Uma instância do problema 3L-TSP é dada por: *(i)* um contêiner de dimensões $B = (W, H, D)$, *(ii)* um grafo $G = (V, E, c)$ com conjunto de vértices $V = \{0, 1, \ldots, n\}$, arestas $E$, custo não negativo $c_e$ para cada aresta $e \in E$ e *(iii)* conjuntos de caixas $L_v$ para cada vértice $v \in V$. O objetivo é obter um circuito hamiltoniano $C = (0, v_1, \ldots, v_n)$ de custo total mínimo e um empacotamento $\mathcal{P}$ de todas as caixas em $B$ que respeite a ordem de $\mathcal{L} = (L_{v_1}, \ldots, L_{v_n})$, denotaremos por $\mathcal{R}_I$ o conjunto das rotas de $I$ que tem empacotamentos que respeitam ordem.

A formulação em programação linear inteira do problema 3L-TSP, que utilizamos na abordagem *branch-and-cut*, é uma formulação do problema TSP, acrescido de restrições que satisfaçam empacotamentos que respeitam ordem. Dada instância $I = (V, E, c, W, H, D, L) \in \mathcal{I}$ o problema 3L-TSP pode ser formulado como:

[1]Esta pesquisa teve o apoio financeiro do CNPq, Fapesp e da CAPES.

$$
\begin{aligned}
&\text{minimize} \sum_{e \in E} c_e x_e \\
&\text{sujeito a} \\
&\sum_{e \in \delta(v)} x_e = 2 \qquad \forall v \in V\,, && (1)\\
&\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset V, \quad S \neq \emptyset\,, && (2)\\
&\sum_{e \in R} x_e \leq n-1 \qquad \forall R \notin \mathcal{R}_I\,, && (3)\\
&x_e \in \{0,1\} \qquad \forall e \in E\,. && (4)
\end{aligned}
$$

As restrições (4) fazem as variáveis $x_e$ para cada $e \in E$ serem binárias e indicam a pertinencia de uma aresta na solução. As restrições (1) e (2) são as restrições básicas da formulação do TSP e garantem que a solução será um circuito hamiltoniano do grafo de entrada. As restrições em (3) restringem o conjunto de rotas apenas para as que podem ter empacotamentos com ordem. Apesar do número de restrições em (2) e (3) ser exponencial, o método *branch-and-cut* insere apenas aquelas necessárias para a resolução. Para isso, alimentamos o algoritmo de *branch-and-bound* com rotinas de separação, que inserem uma restrição violada sempre que detectam que um ponto viola alguma das restrições da formulação.

Como rotinas de separação das restrições (2), utilizamos cortes de conexidade, obtidas a partir de cortes mínimos. Inicialmente, executamos o procedimento da árvore de Gomory-Hu [2, 4], que nos dá todos os cortes mínimos entre vértices em tempo polinomial. Com isso, inserimos todos os cortes violados da árvore. Posteriormente, utilizamos uma heurística que procura por restrições conhecidas como *comb inequalities* [3, 1], desenvolvidas para o TSP. Sempre que um nó inteiro (rota) for encontrado na árvore, executamos as rotinas de separação das restrições (4). Para isso, aplicamos inicialmente as heurísticas e caso nenhuma delas consiga obter um empacotamento que satisfaça a ordem da rota, executamos um algoritmo exato. Caso não haja empacotamento viável que respeite a ordem da rota, a correspondente desigualdade em (4) é inserida e o processo continua.

O algoritmo de Empacotamento exato, denominado por *OneBin* descrito por Martello et al. [6] usa a estratégia *branch-and-bound* e enumera o espaço de busca através das posições onde uma caixa pode ser colocada. Dado um empacotamento parcial $\mathcal{Q}$, as posições de pontos $\rho(\mathcal{Q})$, que são candidatas a receber uma nova caixa são dadas por uma rotina, denominada *3D-Corners*. Tais pontos foram denominados de *pontos de canto*. A complexidade computacional deste algoritmo é $O(m^2)$, onde $m$ é o número de caixas em $\mathcal{Q}$.

O algoritmo exato também pode funcionar como uma heurística, restringindo-se o tempo de sua execução. Assim, denominamos por MPV o algoritmo exato e por MPV $T$, o algoritmo exato limitado a uma execução de $T$ segundos.

A heurísticas que denominamos por GR, é uma adaptação da heurística desenvolvida por George e Robinson [5] que utiliza uma abordagem de *Wall-Building* (construção de parede). Para a obtenção de empacotamentos com ordem, restringimos a ordem das caixas de maneira que todas as caixas de um cliente são empacotadas após as caixas dos clientes anteriores. Com isso, nenhuma caixa empacotada será bloqueada por outra colocada posteriormente. Para uma descrição mais detalhada dessa heurística consulte [5].

A segunda heurística, é baseada no algoritmo híbrido HFF (*Hibrid First Fit*) para o empacotamento de placas retangulares. A heurística, que denominamos HFF3, também empacota as caixas por níveis, permitindo que dois clientes adjacentes compartilhem o mesmo nivel.

Para analisarmos o desempenho do algoritmo usando as heurísticas e o algoritmo exato como geradores de planos de corte e obtenção dos empacotamentos viáveis. Em seguida, analisamos o desempenho do algoritmo quando contrapomos o tamanho do contêiner com o comprimento da rota.

Realizamos testes em instâncias geradas computacionalmente na comparação dos algoritmos. O número de clientes variou entre 7 e 20 e o número de caixas consideradas foi de 9 a 25. Geramos instâncias cujo volume dos itens não passa de 60% do volume do contêiner, denominada de Classe E, e instâncias onde o volume das caixas é de pelo menos 95% do volume do contêiner, denominada de Classe H.

Dependendo das rotinas usadas para a obtenção dos cortes de empacotamento, o algoritmo *branch-and-cut* obtém resultados com tempo e qualidade distintos. Assim, denotaremos por $\mathcal{A}(\mathcal{R})$ o algoritmo *branch-and-cut* usando a(s) rotina(s) de empacotamento $\mathcal{R}$ para gerar cortes de empacotamento.

Na Tabela 1, apresentamos um comparativo das heurísticas e do algoritmo exato na abordagem proposta. As colunas da tabela contém, da esquerda para a direita, as seguintes informações: Classe da instância (*Classe*), número de clientes (*#Clientes*), número total de caixas (*#Caixas*), custo da rota obtida por $\mathcal{A}(\mathrm{GR} + \mathrm{HFF3})$ e seu tempo de processamento, custo da rota obtida por $\mathcal{A}$(MPV10) (MPV10 é igual ao algoritmo MPV, limitado a 10 segundos, por chamada) e seu tempo computacional e custo da rota obtida pelo algoritmo exato $\mathcal{A}(\mathrm{GR} + \mathrm{HFF3} + \mathrm{MPV})$ e seu tempo computacional. No gráfico apresentado acima da tabela 1 custos negativos representam instâncias onde não foi encontrado solução viável.

Apesar de terem um bom desempenho na obtenção de empacotamentos da classe

| *Id* | *Classe* | *#Clientes* | *#Caixas* | 1-Custo | 1-Tempo | 2-Custo | 2-Tempo | 3-Custo | 3-Tempo |
|---|---|---|---|---|---|---|---|---|---|
| 1 | E | 7 | 9 | 2013,51 | 6s | 1644,36 | 0,16s | 1644,36 | 0,11s |
| 2 | E | 7 | 10 | 1709,65 | 0,10s | 1709,65 | 0,10s | 1709,65 | 0,11s |
| 3 | E | 7 | 20 | 1827,16 | 0,14s | 1827,16 | 0,10s | 1827,16 | 0,11s |
| 4 | E | 8 | 15 | 2919,95 | 7s | 2226,04 | 0,12s | 2226,04 | 0,10s |
| 5 | E | 10 | 20 | 3157,46 | 0,14s | 3159,01 | 21s | - | 3600s |
| 6 | E | 10 | 25 | 3704,99 | 21s | 3065,97 | 10s | - | 3600s |
| 7 | H | 10 | 20 | - | 66s | 5575,10 | 11s | 5556,53 | 29s |
| 8 | H | 15 | 20 | - | 108s | 7048,80 | 19s | 7030,24 | 115s |
| 9 | H | 20 | 20 | - | 308s | 7360,54 | 24s | 7310,42 | 177s |
| 10 | H | 20 | 25 | - | 540s | 7366,18 | 231s | 7310,42 | 360s |

Tabela 1: Comparação das soluções heurísticas e exata. 1: $\mathcal{A}(\mathrm{GR} + \mathrm{HFF3})$; 2: $\mathcal{A}$(MPV10); 3: $\mathcal{A}$(GR+HFF3+MPV)

E, as heurísticas GR e HFF3 não obtiveram soluções viáveis para os empacotamentos mais complexos da classe H. Isto fez com que o algoritmo *branch-and-bound* percorresse toda a árvore de enumeração, levando a um grande tempo de processamento para se certificar que de fato não há nenhum ramo onde estas heurísticas conseguem obter empacotamento viável.

Já o algoritmo *branch-and-cut* usando a heurística MPV com limitação de tempo, MPV10, foi mais robusta para restrições de ordem, obtendo soluções viáveis para todas as instâncias, com custo pouco acima do ótimo.

Por ser uma heurística rápida com desempenho razoavelmente bom, analisamos o desempenho do algoritmo $\mathcal{A}$(GR) contrapondo duas funções objetivo: uma que minimiza o custo da rota e outra que minimiza o tamanho do contêiner. A Tabela 2, apresenta o comportamento deste algoritmo para contêiners com largura e altura fixos e o comprimento $D$ variando de 107 a 200, para uma instância de 30 clientes e um total de 120 caixas.

| $D$ | $\mathcal{A}$(GR) | Tempo |
|---|---|---|
| 200 | 14244,49 | 0,66s |
| 150 | 14244,49 | 0,35s |
| 114 | 14244,49 | 50s |
| 113 | 14352,86 | 127s |
| 111 | 14449,47 | 110s |
| 110 | 14449,47 | 83s |
| 109 | 14524,58 | 90s |
| 108 | 24947,90 | 316s |
| 107 | - | 406s |

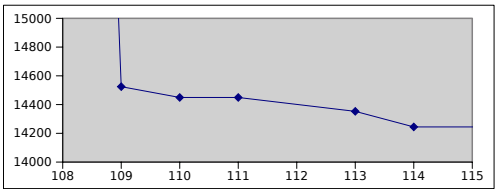


Tabela 2: Desempenho do algoritmo $\mathcal{A}$(GR) contrapondo o comprimento do contêiner com comprimento total das rotas obtidas.

É possível notar que a medida que o tamanho do contêiner diminui, o número de empacotamentos possíveis de serem gerados pela heurística GR que satisfazem a restrição de ordem também diminui, levando muitas vezes a rotas de custo elevado. Por exemplo, na tabela 2, para se usar um contêiner de comprimento 108 (o menor possível), o melhor custo de deslocamento obtido pelo algoritmo $\mathcal{A}$(GR) foi de quase 25000. Isto representa um gasto de deslocamento de quase 72% a mais que a rota usada para um contêiner de tamanho 109.

Por ser rápida, as heurísticas de empacotamento puderem obter soluções viáveis em pouco tempo, e com isso, um usuário pode optar por usar um veículo menor, mas percorrendo uma distância maior, ou um veículo maior, percorrendo menor distância. Certamente uma importante informação na tomada de decisões. Neste trabalho, não utilizamos metaheurísticas nem heurísticas primais para a construção de soluções viáveis para o 3L-TSP. O desenvolvimento de tais heurísticas também será foco de nossa atenção, que além de serem pontos interessantes de investigação, poderão agilizar bastante a busca por soluções ótimas. Nossa intenção é tratar outras condições práticas, como a geração de empacotamentos que respeitam condições de estabilidade e balanceamento; e variantes onde as caixas de um cliente $i$ devem ser carregadas em um ponto de origem $o_i$ e entregues em um ponto de destino $d_i$.

# Referências


[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A computational study*. Princeton Press, 2006.

[2] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *SIAM Journal on Computing*, 9(4):551–570, 1961.

[3] M. Grötschel and M. W. Padberg. On the symmetric travelling salesman problem i: Inequalities. *Mathematical Programming*, 16:265–280, 1979.

[4] D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal on Computing*, 19(1):143–155, 1990.

[5] D. F. Robinson J. A. George. A heuristic for packing boxes into a container. *Computer and Operations Research 7*, 4:147–156, 1980.

[6] S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267, 2000.

[7] F. K. Miyazawa and Y. Wakabayashi. An algorithm for the three-dimensional packing problem with asymptotic performance analysis. *Algorithmica*, 18(1):122–144, 1997.