57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

UM MODELO EXATO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM OPÇÕES DE ENTREGA

Bruno Oliveira
Pós-Graduação em Engenharia de Produção - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Campus Praia Vermelha, Bloco D - sala 309, São Domingos, Niterói-RJ
bruno_mattos@id.uff.br

Artur Pessoa, Diogo Lima, Marcos Roboredo
Departamento de Engenharia de Produção - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Campus Praia Vermelha, Bloco D - sala 306, São Domingos, Niterói-RJ
arturpessoa@id.uff.br, diogofls@id.uff.br, mcroboredo@id.uff.br

RESUMO

Este trabalho aborda o Problema de Roteamento de Veículos com Opções de Entrega, que contempla aspectos relevantes da distribuição na última milha, como a flexibilidade na escolha do local de entrega e o uso compartilhado de pontos de entrega entre clientes. O modelo inclui restrições de capacidades dos veículos e pontos compartilhados, janelas de tempo e atendimento de um índice global de satisfação dos clientes. Para resolver esse problema complexo, este artigo apresenta um algoritmo de *branch-and-cut-and-price*, aplicável de forma direta ou iterativa com base na ferramenta VRPSolver. As abordagens foram avaliadas em instâncias *benchmark* da literatura. Os resultados demonstraram a superioridade da proposta deste artigo em relação ao melhor algoritmo exato disponível até então na literatura, tanto em termos de tempo de execução como em termos de qualidade das soluções.

PALAVRAS CHAVE. Roteamento de Veículos, Distribuição na última Milha, Branch-and-cut-and-price.

Tópicos (Otimização Combinatória, Logística e Transportes)

ABSTRACT

This work addresses the Vehicle Routing Problem with Delivery Options, which captures key aspects of last-mile distribution, such as delivery flexibility and the use of shared delivery locations among customers. The problem formulation includes vehicle and location capacity constraints, time windows, and a global customer satisfaction index. To solve this rich and challenging problem, we present a *branch-and-cut-and-price* algorithm, which can be applied both directly and iteratively, coded within VRPSolver tool. Both approaches were tested on benchmark instances from the literature. The results show a superior performance compared to the best exact algorithm available to date, in terms of both solution quality and computational time.

KEYWORDS. Vehicle Routing, Last-mile delivery, Branch-and-cut-and-price

Paper topics (Combinatorial Optimization, Logistics and Transportation)

https://proceedings.science/p/212479?lang=pt-br

1. Introdução e algoritmo proposto

O foco deste trabalho é o Problema de Roteamento de Veículos com Opções de Entrega (*Vehicle Routing Problem with Delivery Options* — VRPDO). Consideram-se três conjuntos: clientes $C = \{c_1, \ldots, c_{n_c}\}$, locais $L = \{l_0, \ldots, l_{n_l}\}$ e opções de entrega $O = \{o_1, \ldots, o_{n_o}\}$. Cada opção $o_i \in O$ está associada a um cliente $cust(i) \in C$, a um local $loc(i) \in L \setminus \{l_0\}$, a uma janela $[a_i, b_i]$ e a um tempo de serviço s_i. Cada cliente c_j possui demanda $w_j > 0$, a ser entregue em algum local l_k que possua opção o_i com $loc(i) = l_k$ e $cust(i) = c_j$. O depósito l_0 tem janela $[0, b_0]$, onde b_0 é a duração máxima da rota. Cada local l_k possui capacidade cap_k (máximo de clientes atendidos) e tempo de estacionamento $\bar{s}_k$ (com $\bar{s}_0 = 0$). Para cada par de locais l_i, l_j, t_{ij} e d_{ij} denotam tempo e custo de travessia, com matrizes simétricas. Cada cliente atribui prioridade distinta (1 = mais alta) a suas opções. O objetivo é definir rotas que partam e retornem a l_0, visitando opções intermediárias, minimizando hierarquicamente: (i) número de veículos; (ii) custo total. As rotas devem satisfazer: (1) cada cliente tem exatamente uma opção visitada; (2) a soma das demandas na rota não excede Q; (3) o tempo ao visitar o_i deve estar em $[a_i, b_i]$, considerando travessias, serviços e estacionamentos; (4) o tempo total não excede b_0; (5) para cada $i = 1, \ldots, np$, ao menos $\lceil \beta_i \times nc \rceil$ opções com prioridade $\leq i$ devem ser visitadas; (6) em cada local l_k, o número de opções visitadas não excede cap_k; (7) cada local é visitado por no máximo um veículo.

O VRPDO foi originalmente proposto por Tilk et al. [2021], que desenvolveram dois algoritmos exatos baseados em *branch-and-cut-and-price* (BCP) para sua resolução. Os autores adotaram uma abordagem com funções objetivo hierárquicas, priorizando inicialmente a minimização do número de veículos utilizados e, em seguida, a minimização do custo total de travessia. Posteriormente, o problema foi revisitado por Dumez et al. [2021], que propuseram uma metaheurística com o objetivo de resolvê-lo de forma mais eficiente em termos computacionais. Outras variantes também consideram flexibilidade na entrega [Sadati et al., 2022; Duman et al., 2024].

Este trabalho apresenta um modelo exato para o VRPDO, implementado com o auxílio do *VRPSolver* [Pessoa et al., 2020].. Considerando a hierarquia entre as funções objetivo, demonstramos que o algoritmo pode ser implementado também de forma iterativa. Ambas as versões propostas do algoritmo são testadas em instâncias *benchmark* do problema e comparadas com o melhor algoritmo exato disponível na literatura, proposto por Tilk et al. [2021]. Os resultados mostram que os algoritmos desenvolvidos superam a abordagem de referência tanto em termos de tempo computacional quanto no número de soluções comprovadamente ótimas obtidas.

O algoritmo BCP utilizado é gerado no *framework* VRPSolver, que exige a definição de uma formulação por partição de conjunto baseada em variáveis de caminho sobre um ou mais grafos direcionados. Nessas estruturas, o usuário pode associar recursos que possuem consumo em cada nó do grafo, os quais são acumulados durante a construção dos caminhos. Um caminho é considerado inviável caso o consumo de algum recurso ultrapasse um limite previamente especificado. A modelagem via recursos permite a representação eficiente de diversas restrições do problema, como capacidade dos veículos, janelas de tempo, entre outras.

1.1. Formulação por partição de conjunto

Seja Ω o conjunto de rotas (caminhos) que começam e terminam no depósito e que respeitam a restrição de capacidade dos veículos e as restrições de janela de tempo. Seja ainda o grafo $G = (V, E)$ tal que $V = \{0, 1, 2, \ldots, no\}$ e $E = \{(i, j) : i \in V, j \in V, i < j\}$. Com base nestes conjuntos, a formulação por partição de conjunto considera as seguintes variáveis: λ_r é binária e indica se uma rota $r \in \Omega$ é usada; x_e é inteira e indica o número de vezes que a aresta $e \in E$ é usada; y_k é binária e indica se o local l_k é visitado; z é inteira e indica o número de veículos usados. Além disso, consideram-se as constantes: h_{er} indica quantas vezes a aresta $e \in E$ é usada na rota $r \in \Omega$; h'_{kr} indica quantas vezes o local l_k é visitado na rota $r \in \Omega$; $\delta(c_k) = \{(i, j) \in$

$E : \text{cust}(i) = c_k$ ou $\text{cust}(j) = c_k\}$; $\delta'(l_k) = \{(i,j) \in E : \text{loc}(i) = l_k$ ou $\text{loc}(j) = l_k\}$. A função $\alpha_l(e = (i,j), l_k)$ vale 0 se $\text{loc}(i) \neq l_k$ e $\text{loc}(j) \neq l_k$, 2 se $\text{loc}(i) = l_k$ e $\text{loc}(j) = l_k$, e 1 caso contrário. A função $\alpha_p(e = (i,j), k)$ vale 0 se os índices de prioridade de i e j são ambos maiores que k, 2 se ambos são menores ou iguais a k, e 1 caso contrário. Por fim, M é um número suficientemente grande para garantir a hierarquia das funções objetivo.

$$
\begin{aligned}
&\text{Min } Mz + \sum_{e \in E} d_e x_e \\
&\text{S.a. } \sum_{e \in \delta(c_k)} x_e = 2, && k = 1, \ldots, nc; && \sum_{e \in \delta'(l_k)} x_e = 2y_k, && \forall k = 1, \ldots, nl \\
&\quad \sum_{e \in E} \alpha_l(e, l_k) x_e \leq 2cap_k && k = 1, \ldots, nl; && \sum_{e \in E} \alpha_p(e,k) x_e \geq 2 \lceil \beta_k \times nc \rceil && k = 1, \ldots, np \\
&\quad x_e = \sum_{r \in \Omega} h_{er} \lambda_r && e \in E; \\
&\quad y_k = \sum_{r \in \Omega} h'_{kr} \lambda_r && k = 1, \ldots, nl \\
&\quad z = \sum_{r \in \Omega} \lambda_r
\end{aligned}
$$

1.2. Grafo gerador de caminhos

Para que o VRPSolver possa ser utilizado, é necessário definir um grafo direcionado para gerar os caminhos em Ω. Seja $\bar{G} = (V, A)$, onde V foi definido anteriormente e $A = \{(i,j), (j,i) : (i,j) \in E\}$. Seja p' um caminho neste grafo que começa e termina no vértice 0. Associados a p', definimos os consumos $S^1_{p'}$ e $S^2_{p'}$ para garantir as restrições de capacidade e janelas de tempo. Inicialmente, $S^1_{p'} = S^2_{p'} = 0$. Quando um arco $a = (i,j)$ é percorrido em p', temos: $S^1_{p'}$ é incrementado por $w_{cust(j)}$ se $j \in O$ e permanece inalterado se $j = l_0$; já $S^2_{p'}$ é atualizado da seguinte forma: se $i = l_0$, então $S^2_{p'} = \max\{S^2_{p'} + t_{0j} + \bar{s}_{loc(j)}, a_j\}$; se $i, j \in O$ e $\text{loc}(i) \neq \text{loc}(j)$, então $S^2_{p'} = \max\{S^2_{p'} + s_i + t_{ij} + \bar{s}_{loc(j)}, a_j\}$; se $i, j \in O$ e $\text{loc}(i) = \text{loc}(j)$, então $S^2_{p'} = \max\{S^2_{p'} + s_i + t_{ij}, a_j\}$; e se $i \in O$ e $j = l_0$, então $S^2_{p'} = S^2_{p'} + s_i + t_{0j}$. Notemos que $S^1_{p'}$ representa a demanda total atendida após visitar a opção de entrega j, enquanto $S^2_{p'}$ representa o tempo de chegada nessa opção. O caminho p' é considerado inviável se $S^1_{p'} > Q$ ou $S^2_{p'} > b_i$, sendo assim descartado como variável de rota válida na Formulação (1).

A Formulação e grafo descritos respectivamente nas Seções 1.1 e 1.2 são fornecidos ao ferramenta computacional VRPSolver, proposta por Pessoa et al. [2020]. A ferramenta aplica então um BCP genérico que considera a Formulação 1 apenas com variáveis λ (as variáveis x e z são substituídas as através das três últimas restrições apresentadas). A vantagem do uso da ferramenta VRPSolver é que o algoritmo BCP genérico gerado pela ferramenta já possui diversas técnicas estado-da-arte.

2. Resultados

Uma vez que a função objetivo prioritária é a minimização do número de veículos, aplicamos o algorimo de maneira iterativa, resolvendo o problema fixando o número de veículos inicialmente em $K_{lb} = \lceil \sum w_j / Q \rceil$. Caso o problema se mostre inviável aumentamos em unidade o valor de K_{lb} em uma unidade. O procedimento continua até que a viabilidade seja alcançada.

Todos os experimentos foram realizados em um computador equipado com um processador Intel Core i7-10700 com *clock* de 2,90 GHz e 16 GB de RAM. O sistema operacional utilizado foi o Ubuntu 18.04, e o algoritmo BCP foi gerado a partir do *framework* VRPSolver v0.4.1a (`https://vrpsolver.math.u-bordeaux.fr/`). As instâncias utilizadas para os testes computacionais foram propostas por Tilk et al. [2021]. Nos referimos ao artigo prévio para uma descrição detalhada das mesmas.

https://proceedings.science/p/212479?lang=pt-br

Grupo de Instâncias	Tempo médio(s)			#opt		
	Este trabalho			Este trabalho		
	Alg. Iterativo	Alg. simples	Lit.	Alg. Iterativo	Alg. simples	Lit.
V25_pequena	22,1	143,9	23,7	10	10	10
V25_média	51,8	478,5	342,3	10	10	10
V25_grande	121,6	642,8	656,2	10	10	10
U25_pequena	7,1	73,1	16,5	10	10	10
U25_média	471,8	1347,2	761,8	10	9	9
U25_grande	127,6	893,9	1115,4	10	10	9
V50_pequena	507,6	1923,5	2860,7	10	8	6
V50_média	334,7	1651,7	3399,8	10	9	4
V50_grande	1334,7	2925,6	4355,0	8	6	2
U50_pequena	276,2	73,1	2811,6	10	10	6
U50_média	1455,6	1347,2	4905,1	9	9	1
U50_grande	1924,6	893,9	4587,0	7	10	1
Global	552,9	1032,9	2152,9	9,5	9,3	6,5

Tabela 1: Comparação com a literautura

A Tabela 1 apresenta uma comparação entre as abordagens propostas neste trabalho — o Algoritmo Iterativo, indicado na coluna *Alg. Iterativo*, e o modelo VRPSolver proposto, indicado na coluna *Alg. Simples* — e o melhor algoritmo exato da literatura, proposto por Tilk et al. [2021], indicado na coluna *Lit.*. A comparação considera dois critérios: o tempo médio de execução (coluna *Tempo médio (s)*) e o número de soluções comprovadamente ótimas encontradas (coluna *#opt*) para cada grupo de 10 instâncias. Essas instâncias variam de acordo com o tipo (U ou V), o número de clientes (25 ou 50), e a largura das janelas de tempo (pequena, média ou grande). A partir da Tabela 1, é possível concluir que o algoritmo iterativo proposto supera consistentemente o estado da arte em todos os grupos de instâncias, tanto em termos de tempo médio de execução quanto no número de soluções comprovadamente ótimas obtidas.

3. Conclusão

Este apresentou um algoritmo BCP exato, implementado com o auxílio do VRPSolver, que pode ser executado tanto de maneira direta quanto iterativamente. Ambas abordagens foram testadas em instâncias *benchmark* da literatura, onde foi possível constatar que estas superam o melhor algoritmo exato existente até então.

Referências

Duman, E. N., Taş, D., e Çatay, B. (2024). A bidirectional branch-and-price algorithm with pulse procedure for the electric vehicle routing problem with flexible deliveries. *Transportation Research Part C: Emerging Technologies*, 165:104699.

Dumez, D., Lehuédé, F., e Péton, O. (2021). A large neighborhood search approach to the vehicle routing problem with delivery options. *Transportation Research Part B: Methodological*, 144: 103–132.

Pessoa, A., Sadykov, R., Uchoa, E., e Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183:483–523.

Sadati, M. E. H., Akbari, V., e Çatay, B. (2022). Electric vehicle routing problem with flexible deliveries. *International Journal of Production Research*, 60(13):4268–4294.

Tilk, C., Olkis, K., e Irnich, S. (2021). The last-mile vehicle routing problem with delivery options. *Or Spectrum*, 43(4):877–904.