LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

UM MODELO EXATO PARA UM PROBLEMA DO ROTEAMENTO DE VEÍCULOS COM LOCKERS

Bruno Davi Mattos de Oliveira
Programa de 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

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

RESUMO

Os Problemas de Roteamento de Veículos com Lockers (PRVL) envolvem estruturas de armazenamento de produtos localizadas estrategicamente para receber demandas de diferentes clientes. Clientes atribuídos a *lockers* devem se locomover até o local atribuído, gerando um custo de atribuição. O objetivo é, assim, definir rotas e atribuições de modo que todo cliente seja atendido diretamente ou atribuído a algum *locker* visitado, visando minimizar a soma dos custos de travessia e atribuição. Este estudo examina uma variante que inclui janelas de tempo, capacidades nos veículos e nos *lockers*, e a limitação de uma única visita a cada *locker*. Para resolvermos o problema de forma exata, propomos um algoritmo de Branch-and-Cut-and-Price utilizando o framework VRPSolver. Experimentos computacionais demonstram a robustez do procedimento, que supera o melhor método exato da literatura e obtém a solução ótima de diversas instâncias com até 60 clientes pela primeira vez.

PALAVRAS CHAVE. Roteamento de veículos, *Lockers*, Janelas de tempo, *Branch-cut-and-price*

Tópicos (Otimização Combinatória, Logística e Transportes)

ABSTRACT

The Vehicle Routing Problems with Lockers (VRPL) involve strategically located storage structures to receive demands from different customers. Customers assigned to lockers must travel to the designated location, incurring an assignment cost. The objective is to define routes and assignments so that every customer is either directly served or assigned to a visited locker, aiming to minimize the total traversal and assignment costs. This study examines a variant that includes time windows, vehicle and locker capacities, and the limitation of a single visit to each locker. To solve the problem exactly, a Branch-and-Cut-and-Price algorithm is proposed using the VRPSolver framework. Computational experiments demonstrate the robustness of the procedure, which outperforms the best exact method in the literature and achieves the optimal solution for several instances with up to 60 customers for the first time.

KEYWORDS. Vehicle routing, Lockers, Time Windows, Branch-cut-and-price

Paper topics (Combinatorial Optimization, Logistics and Transportation)

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

1. Introdução

Desde Dantzig e Ramser [1959], os Problemas de Roteamento de Veículos (PVRs), originalmente chamados de *Vehicle Routing Problems* (VRPs), têm se destacado entre as aplicações práticas da Pesquisa Operacional, envolvendo vários desafios nas atividades diárias das organizações. Frequentemente, características práticas de problemas de roteamento são incorporadas aos modelos de forma que estes abordem aspectos reais enfrentados por gerentes e tomadores de decisão. Por exemplo, problemas específicos de roteamento incluem frotas heterogêneas, janelas de tempo, seleção de hotéis e paradas para reabastecimento de veículos.

Uma classe de PRVs que vem chamando a atenção de pesquisadores nos últimos anos é aquela que envolve os chamados *lockers*, que são estruturas especiais, localizadas estrategicamente, onde as demandas de diferentes clientes são armazenadas. Os clientes então se deslocam até o *locker* onde sua demanda foi armazenada para a sua retirada, o que gera um custo de atribuição. Assim, nos PRVs com *lockers*, deve-se definir as rotas de cada veículo, bem como decidir quais clientes são diretamente atendidos e quais são atribuídos a algum *locker* de modo que o custo de travessia somado ao custo de atribuição seja o menor possível. Dentre os trabalhos que focam em PRVs, destacam-se os propostos por Grabenschweiger et al. [2021], Vincent et al. [2022], Buzzega e Novellani [2023], Dell'Amico et al. [2023] e Boschetti e Novellani [2024].

Neste estudo, focamos na variante PRV com *lockers* proposta por Buzzega e Novellani [2023], que possui as seguintes características: a demanda de cada cliente pode ser atendida diretamente na casa do cliente ou ser armazenada em algum *locker* visitado próximo ao cliente; tanto os veículos quanto os *lockers* possuem capacidades; e cada *locker* só pode ser visitado por no máximo um veículo. Os autores consideraram duas variantes, com e sem janelas de tempo nos vértices, aqui referidas respectivamente como PRVLJT e PRVL. Para resolver o problema de forma exata, Buzzega e Novellani [2023] propuseram formulações de Programação Linear Inteira Mista (PLIM).

Conforme destacado por Costa et al. [2019], dentre as formas de resolver PRVs de maneira exata, destaca-se os algoritmos *Branch-Cut-and-Price* (BCP). Infelizmente, a implementação de um algoritmo BCP com componentes estado-da-arte para um VRP específico requer muito tempo, mesmo para uma equipe habilidosa. Para mitigar esse problema, Pessoa et al. [2020] propuseram o *framework* VRPSolver, capaz de gerar um algoritmo BCP para resolver um VRP específico baseado na modelagem fornecida pelo usuário. Um modelo VRPSolver contém uma formulação PLIM que inclui variáveis representando rotas viáveis. O conjunto de rotas viáveis é modelado por um conjunto de caminhos com restrições de recursos em um ou vários grafos direcionados. O VRPSolver resolve o modelo usando um algoritmo BCP genérico que gera variáveis de caminho dinamicamente, resolvendo os subproblemas de *pricing*. Estes são modelados no *framework* como problemas do caminho mais curto com restrições de recurso. Além da formulação PLIM e dos grafos direcionados, o usuário também pode definir os chamados *packing sets*, que permitem ao usuário ativar alguns componentes estado-da-arte para algoritmos BCP, tais como *NG paths*, geração de cortes de capacidade arredondados e de cortes de rank-1 com memória limitada, enumeração de rotas, entre outras. Algoritmos BCP gerados com auxílio do VRPSolver são competitivos ou até mesmo considerados o estado-da-arte para diversos PRVs [Roboredo et al., 2023; Soares e Roboredo, 2023; Praxedes et al., 2024].

Este trabalho propõe um algoritmo BCP exato para o PRVLJT e PRVL gerado a partir do VRPSolver. Como sua principal característica, a modelagem proposta não considera todas as possibilidades de atribuição no subproblema de *pricing*. Ao invés disso, considera-se no subproblema de *pricing* qual a demanda total atribuída a um *locker*, mas a decisão de quais clientes são efetivamente alocados é tomada no problema mestre. Para demonstrar a robustez do método proposto, apresentamos diversos experimentos computacionais nas 60 instâncias com até 60 clientes também usadas

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

por Buzzega e Novellani [2023]. Resultados mostram que o método proposto supera a literatura, uma vez que apresenta tempo computacional menor para todas as instâncias e resolve diversas destas até otimalidade pela primeira vez. Apresentamos ainda o custo ótimo de todas as 60 instâncias em tempos computacionais razoáveis.

2. O Problema

O PRVLJT é definido sobre um grafo não dirigido completo $G = (V, E)$, onde o conjunto de vértices V é particionado em $V = \{0\} \cup V_C \cup V_L$; onde 0 indica o depósito, $V_C = \{1, 2, ..., n\}$ representa um conjunto de n clientes e $V_L = \{n+1, n+2, ..., n+m\}$ representa um conjunto de m *lockers* disponíveis. Cada vértice $i \in V$ está associado a uma janela de tempo $[a_i, b_i]$. Cada cliente $i \in V_C$ está associado a uma demanda positiva d_i e, para cada aresta $e \in E$, está associado um custo e a um tempo de travessia positivos, denotados por c_e e t_e, respectivamente. Para o atendimento dos clientes, existe uma frota não limitada de veículos homogêneos, onde cada um destes está associado a uma capacidade positiva Q. Para ser atendido, cada cliente $i \in V_C$ pode ser visitado diretamente por uma das rotas, ou pode ser atribuído a algum *locker* $l \in V_L$ visitado por alguma das rotas, o que gera um custo de atribuição c'_{il}. Cada *locker* $l \in V_L$ possui uma capacidade de atendimento, ν_l, e um raio de cobertura, R_l. Seja γ_l o conjunto de clientes que podem ser atendidos pelo *locker* $l \in V_L$.

O objetivo é definir rotas que começam e terminam no depósito, bem como definir atribuições de clientes a *lockers* de modo a respeitar as seguintes restrições:

- Todo cliente $i \in V_C$ deve ser diretamente visitado por uma das rotas ou deve ser atribuído a algum *locker* $l \in V_l$ visitado por uma das rotas tal que $c_{(i,l)} \leq R_l$.
- Nenhum *locker* pode ser visitado mais de uma vez mesmo que por veículos diferentes.
- O número de clientes atribuídos a um *locker* $l \in V_L$ não excede ν_l.
- Para cada veículo, a soma das demandas dos clientes visitados diretamente por ele com a soma das demandas dos clientes atribuídos a *lockers* visitados por ele não pode ultrapassar a capacidade Q.
- Um vértice $i \in V$ só pode ser visitado dentro da sua janela de tempo $[a_i, b_i]$, incluindo o depósito. Tal imposição não é considerada na versão do problema sem janelas de tempo (VRPL).

As decisões de roteamento e atribuição são feitas de modo a minimizar a soma do custo total de travessia com o custo total de atribuição.

Para exemplificar a situação descrita acima, considere o Exemplo Ilustrativo 1, com $n = 4$, $m = 2$, $Q = 3$, $\nu_5 = \nu_6 = 2$. Considere que os raios dos *lockers* são definidos de tal forma que $\gamma_5 = \{1, 2\}$ e $\gamma_6 = \{2, 3\}$. Além disso, considere que as demandas são unitárias. Na Figura 1, onde clientes são representados por círculos e *lockers* por triângulos, ilustra-se uma instância com essas características e três exemplos de soluções viáveis: Sol_1 = {Rotas:{$(0 \to 1 \to 2 \to 0), (0 \to 4 \to 3 \to 0)$}; Atribuições:$\emptyset$}, Sol_2 = {Rotas:{$(0 \to 5 \to 0), (0 \to 4 \to 3 \to 0)$}; Atribuições:{$(1, 5), (2, 5)$}} e Sol_3 = {Rotas:{$(0 \to 4 \to 0), (0 \to 5 \to 6 \to 0)$}; Atribuições:{$(1, 5), (2, 6), (3, 6)$}}. Perceba que enquanto a primeira solução não realiza atribuições e constrói duas rotas que utilizam duas unidades de capacidade. No segundo exemplo, uma das rotas visita o *locker* 5 para o qual são designados os atendimentos dos clientes 1 e 2. Destaca-se que esta rota, apesar de visitar apenas um *locker*, também consome duas unidades de capacidade do veículo (dois clientes são atendidos). No terceiro exemplo, uma das rotas visita

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

apenas o cliente 4 e retorna ao depósito enquanto a outra rota visita os dois armários e consumiu as três unidades de capacidade do veículo. Percebe-se também que é inviável atender todos os clientes com um único veículo uma vez que excederia-se a restrição de capacidade.

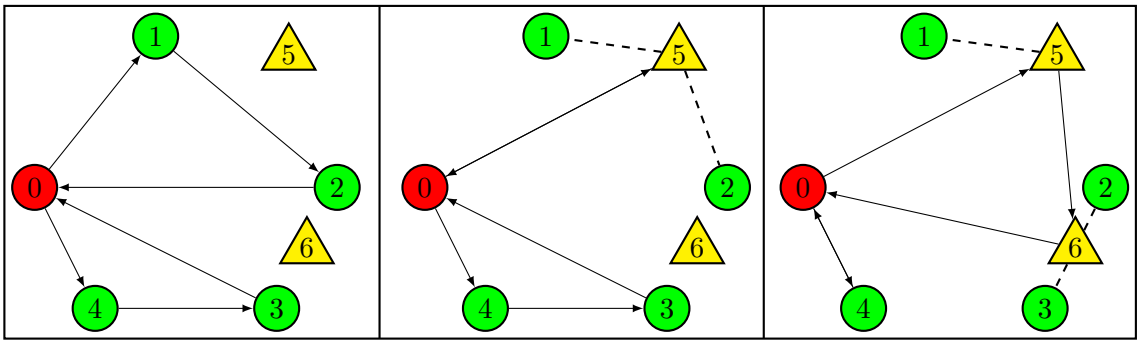

Figura 1: Ilustração do grafo para o exemplo ilustrativo.

3. Modelo VRPSolver proposto para o PRVL

Um modelo VRPSolver consiste em um modelo PLIM que também inclui variáveis de caminho em um ou mais grafos direcionados definidos pelo usuário. O problema é resolvido por meio de um algoritmo genérico BCP onde as variáveis de caminho são geradas sob demanda através da resolução dos subproblemas de *pricing*, modelados como Problemas de Caminho Mais Curto com Restrição de Recursos (RCSPP). As Seções 3.1 e 3.2 apresentam, respectivamente, o grafo gerador de caminhos e a formulação que compõem o modelo VRPSolver proposto para o PRVLJT.

3.1. Grafo gerador de caminhos

Seja $G' = (V', A)$ o grafo gerador de caminhos restritos em recursos definido para o subproblema do RCSPP, e sejam v_{so} e v_{si} vértices desse grafo que representam respectivamente a fonte (*source*, nó inicial) e o semidouro (*sink*, nó final) dos caminhos. O conjunto de vértices é dado por $V' = \{v_{so}, v_{si}\} \cup \{v_i | i \in V \setminus \{0\}\} \cup \{\bar{v}_l | l \in V_L\}$ e $A = A_1 \cup A_2 \cup A_3 \cup A_4 \cup A_5$, onde:

- $A_1 = \{(v_{so}, v_i), (v_{so}, v_l), (v_i, v_{si}), (\bar{v}_l, v_{si}), \, | \, i \in V_C, \, l \in V_L\}$
- $A_2 = \{(v_i, v_j), (v_j, v_i) \, | \, i, j \in V_C, \, i \neq j\}$
- $A_3 = \{(v_i, v_l) \, | \, i \in V_C, l \in V_L\}$
- $A_4 = \{(\bar{v}_l, v_i), \, | \, l \in V_L, \, i \in V \setminus v_l\}$
- $A_5 = \{(v_l, \bar{v}_l, k) \, | \, l \in V_L, k = 1, ..., Q\}$

A ideia por trás do grafo é que, cada rota viável do PRVLJT possa ser representada por um caminho em G' começando em v_{so} e terminando em v_{si}. Assim, estes vértices representem o depósito respectivamente no início e no fim da rota, v_i, com $i \in V_C$ representam o clientes i, e v_l e $\bar{v}_l$, com $l \in V_L$ representam o *locker* l na entrada e na saída dos veículos, respectivamente. Assim, o conjunto de arcos de A_1 representam as possíveis chegadas e saídas no depósito, os arcos A_2 representam conexões entre clientes, os arcos de A_3 e A_4 representem respectivamente a chegada e saída nos *lockers*. O conjunto de arcos A_5 representam conexões paralelas entre um vértice que represente a entrada em um *locker* l (v_l) e um vértice que represente a sua saída ($\bar{v}_l$), que são diferenciados por um terceiro índice k, $k \in \{1, ..., Q\}$. O objetivo é que, quando um arco $(v_l, \bar{v}_l, k)$

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

é percorrido, isso indique que o *locker* está sendo visitado e uma demanda total de k vai ser atribuída no mesmo.

Para se garantir que os caminhos estejam associados a rotas que respeitem a capacidade do veículo e as janelas de tempo dos vértices, são criados dois recursos 1 e 2, onde cada arco $a = (i, j) \in A$ esteja associado aos seguintes consumos q_a^1 e q_a^2 dos recursos 1 e 2, respectivamente, da seguinte maneira:

$$q_a^1 = \begin{cases} d_j \text{ se } a = (i, v_j)\,,\ i \in V',\ j \in V_C \\ k \text{ se } a = (v_l, \bar{v}_l, k)\,,\ l \in V_L \\ 0, \text{ caso contrário.} \end{cases}$$

$$q_a^2 = \begin{cases} t_a \text{ se } a \in A \setminus A_5 \\ 0 \text{ se } a \in A_5 \end{cases}$$

Seja p um caminho sobre o grafo G' começando e terminando nos vértices v_{so} e v_{si}, respectivamente. Seja ainda um vértice $j \in V'$ qualquer que é visitado no caminho p. Denotamos o consumo acumulado dos recursos 1 e 2 no caminho p ao visitar o vértice j como respectivamente $S_{j,p}^1$ e $S_{j,p}^2$ e definimos o seu cálculo da seguinte maneira. $S_{j,p}^1 = S_{j,p}^2 = 0$ quando $j = v_{so}$. Para o caso de $j \neq v_{so}$, seja (i, j) o arco de incidência em j no caminho p. Para recurso 1, definimos $S_{j,p}^1 = S_{i,p}^1 + q_{(i,j)}^1$. Já para o recurso 2, definimos $S_{j,p}^2 = max\{S_{i,p}^2 + q_{(i,j)}^2, a(j)\}$, onde $a(j)$ representa o início da janela de tempo do vértice no grafo original G representado pelo vértice j. Dizemos que p em G' é viável se $S_{j,p}^1 \leq Q$ e $a(j) \leq S_{j,p}^2 \leq Q$, para todo $j \in p$.

Para ilustração de G', considere uma pequena instância com $n = 2$, $m = 1$, $Q = 2$, $T = 3$ $\nu_3 = 2$, $\gamma_3 = \{1, 2\}$. Para cada aresta $e = \{i, j\} \in E, t_{\{i,j\}} = t_{\{j,i\}}$, e sejam esses tempos de travessia dados por $t_{\{0,1\}} = t_{\{0,2\}} = t_{\{0,3\}} = t_{\{1,3\}} = t_{\{2,3\}} = 1$ e $t_{\{1,2\}} = 2$. Por fim, considere demandas unitárias para os clientes, $d_1 = d_2 = 1$. A Figura 2 apresenta o grafo gerador de caminhos restritos em recurso para esse exemplo, onde os consumos de recurso estão ilustrados nos arcos do grafo e as janelas de tempo dos vértices estão representadas em colchetes. Note que dois arcos são criados entre os vértices v_3 e $\bar{v}_3$, indicando que existem duas opções de alocação de demanda neste *locker* caso este seja utilizado.

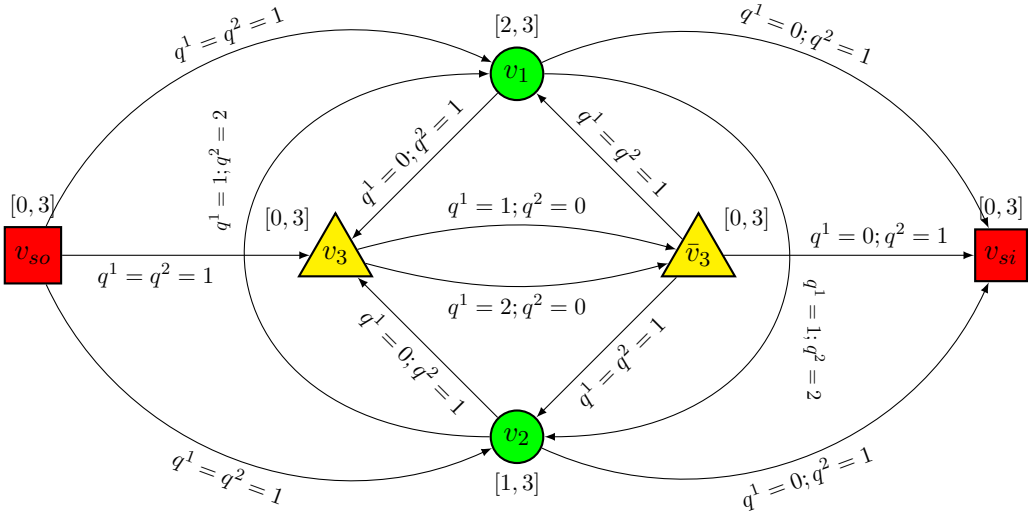

Figura 2: Grafo gerador de caminhos restritos em recurso para o Exemplo Ilustrativo 2 - consumo

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

3.2. Formulação PLIM

Apresentamos agora a formulação que é utilizada pelo modelo VRPSolver proposto. Tal formulação usa as seguintes constantes: o conjunto P representa o conjunto de caminhos viáveis em G' e, para cada *locker* $l \in V_L$, definimos C_l como o conjunto de clientes que podem ser atribuídos a l. Matematicamente, temos $C_l = \{j \in V_C | c_{(j,l)} \leq R_l\}$. A formulação proposta considera as seguintes variáveis. Para cada $e \in E$, definimos uma variável inteira x_e indicando o número de vezes que a aresta e é percorrida. Para cada *locker* $l \in L$, para cada cliente $j \in C_l$, definimos a variável binária y_{jl} indicando se o cliente j é atribuído ao *locker* l (y_{jl} = 1) ou não (y_{jl}=0).Para cada *locker* $l \in V_L$, define-se a variável binária t_l indicando se o *locker* l é visitado (t_l = 1) ou não (t_l=0). Para cada *locker* l e cada possível valor de demanda $k = 1, ..., Q$, definimos a variável z_l^k indicando a demanda total atribuída ao *locker* l é igual a k (z_l^k = 1) ou não (z_l^k=0). Finalmente, para cada caminho $p \in P$, definimos a variável inteira λ_p indicando quantas vezes a rota correspondente ao caminho p é usada na solução. Para relacionar as variáveis de caminho.

A formulação proposta é apresentada no modelo (1).

$$\text{Min} \sum_{e \in E} c_e x_e + \sum_{l \in V_L} \sum_{i \in \mathcal{C}_l} c'_{il} y_{il} \tag{1a}$$

$$\text{S.a.} \sum_{e \in \delta(\{j\})} x_e + \sum_{l\,:\,j \in \mathcal{C}_l} 2y_{jl} = 2 \qquad \forall j \in V_C; \tag{1b}$$

$$\sum_{e \in \delta(\{l\})} x_e = 2t_l \qquad \forall l \in V_L; \tag{1c}$$

$$y_{il} \leq t_l \qquad \forall l \in V_L\,,\, i \in \mathcal{C}_l; \tag{1d}$$

$$\sum_{j \in C_l} y_{jl} \leq \nu_l \qquad \forall l \in V_L; \tag{1e}$$

$$\sum_{k=1}^{Q} k z_l^k = \sum_{i \in \mathcal{C}_l} d_i y_{il} \qquad \forall l \in V_L; \tag{1f}$$

$$x_e = \sum_{p \in P} \left(\sum_{a \in M(x_e)} \alpha_a^p \right) \lambda_p \qquad \forall e \in E; \tag{1g}$$

$$t_l = \sum_{p \in P} \left(\sum_{a \in M(t_l)} \alpha_a^p \right) \lambda_p \qquad \forall l \in V_L; \tag{1h}$$

$$z_l^k = \sum_{p \in P} \left(\sum_{a \in M(z_l)} \alpha_a^p \right) \lambda_p \qquad \forall l \in V_L, k \in \mathcal{D}_l; \tag{1i}$$

$$0 \leq \sum_{p \in P} \lambda_p \leq |V_C|; \tag{1j}$$

$$x_e \in \mathbb{Z}_+ \qquad \forall e \in E; \tag{1k}$$

$$y_{il} \in \mathbb{B} \qquad \forall l \in V_L, i \in \mathcal{C}_l; \tag{1l}$$

$$z_l^k \in \mathbb{B} \qquad \forall l \in V_L, k \in \mathcal{D}_l; \tag{1m}$$

$$t_l \in \mathbb{B} \qquad \forall l \in V_L; \tag{1n}$$

$$\lambda_p \in \mathbb{Z}_+ \qquad \forall p \in P. \tag{1o}$$

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

A função objetivo (1a) visa minimizar a soma do custo total de travessia com o custo total de atribuição. As restrições (1b) garantem que todo cliente $j \in V_C$ deve ser atendido ou em casa ou através de alguma atribuição. Quando $\sum_{e \in \delta(\{j\})} x_e = 2$, o cliente j está sendo atendido em casa. Já quando $\sum_{l \in \mathcal{L}_j} 2y_{jl} = 2$, temos que j está sendo atendido via uma atribuição. As restrições (1c) garantem o grau de um locker $l \in V_L$ é igual a 2 se este locker é visitado ($t_l = 1$) ou é igual a 0 se este locker não é visitado ($t_l = 0$). Restrições (1d) reforçam que para um cliente ser alocado a um locker, esse locker deve ser visitado. Restrições (1e) garantem que a demanda atribuída a um locker não ultrapassa a sua capacidade. Restrições (1f) asseguram que todos os lockers satisfaçam as demandas alocadas a eles. Restrições (1g), (1h) e (1i) relacionam respectivamente as variáveis x, y e z com as variáveis λ através da chamada função de mapeamento M definida da seguinte forma para as variáveis x:

$$M(x_{(i,j)}) = \begin{cases} \{(v_{so}, v_j), (v_j, v_{si})\} \text{ se } i = 0, j \in V_C \\ \{(v_{so}, v_j), (\bar{v}_j, v_{si})\} \text{ se } i = 0, j \in V_L \\ \{(v_i, v_j), (v_j, v_i)\} \text{ se } i, j \in V_C \\ \{(v_i, v_j), (\bar{v}_j, v_i)\} \text{ se } i \in V_C, j \in V_L \\ \{(\bar{v}_i, v_j), (\bar{v}_j, v_i)\} \text{ se } i, j \in V_L \end{cases}$$

Para as variáveis t e z, definimos respectivamente $M(t_l) = \{a = (v_l, \bar{v}_l, k) \,|\, k = 1, ...Q\}$ e $M(z_l^k) = \{a = (v_l, \bar{v}_l, k)\}$. Basicamente, o valor de cada variável x, y ou z é dado pelo número de vezes que os arcos pertencentes aos seus respectivos conjuntos de mapeamentos $M(x)$, $M(y)$ e $M(z)$ são usados pelos caminhos da solução. Para as variáveis y, não houve a necessidade da definição de uma função de mapeamento. Restrição (1j) apresenta um limite inferior e superior para o número de caminhos (rotas) usados na solução. As demais restrições garantem o domínio das variáveis.

A Figura 3 ilustra a função de mapeamento das variáveis x, t e z no grafo gerador de caminhos onde, sobre cada arco do grafo, colocamos as variáveis para as quais este arco faz parte do conjunto de mapeamento.

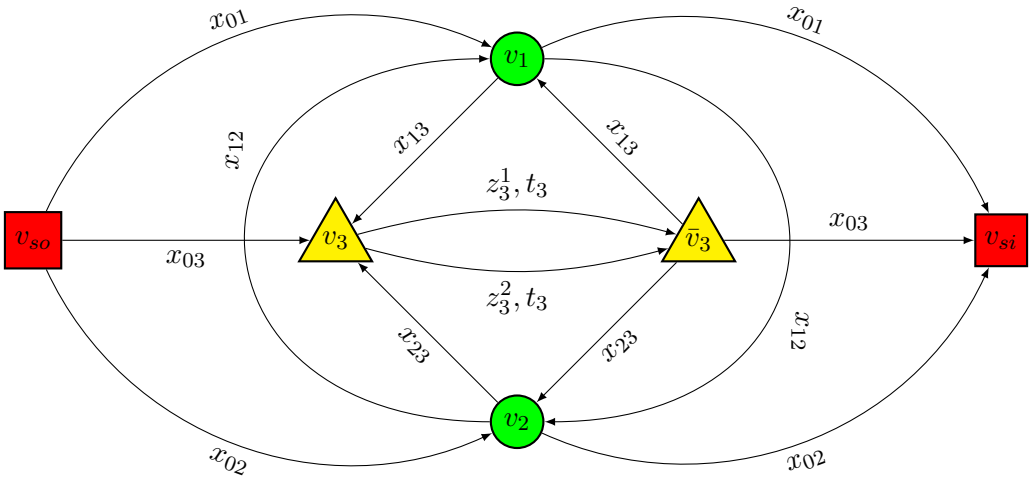

Figura 3: Grafo gerador de caminhos restritos em recurso para o Exemplo Ilustrativo 2 - mapeamento

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

A partir do grafo gerador de caminhos e da formulação PLIM definidos anteriormente, o VRPSolver aplica um algoritmo BCP genérico. Para que componentes estado-da-arte para algoritmos BCP sejam ativadas, é necessário definir os chamados *packing sets*, que são subconjuntos de vértices do grafo G' para os quais no máximo um de seus elementos pode pertencer a no máximo um único caminho na solução ótima. Como tanto os clientes e os lockers só podem ser visitados uma única vez de acordo com a definição do problema, cada conjunto unitário $\{v_i\}$, $i \in V_C \cup V_L$ e cada conjunto unitário $\{\bar{v}_l\}$ são definidos como um packing set. Para mais detalhes a respeito do algoritmo BCP genérico, sugerimos a leitura de Pessoa et al. [2020].

4. Experimentos Computacionais

Nesta seção, apresentamos diversos experimentos computacionais para demonstrar a robustez do modelo proposto. Todos os experimentos foram realizadas em um computador com um processador Intel Core i7-10700 com 2.90 GHz e 16 GB de memória RAM. O sistema operacional usado é o Ubuntu e o algoritmo foi implementado em Julia v1.4.2 usando o framework VRPSolver v0.4.1a (`https://vrpsolver.math.u-bordeaux.fr/`).

Nós executamos cada uma das instâncias testadas neste trabalho de duas maneiras. Na primeira delas, nós testamos o modelo proposto sem o fornecimento de nenhum limite superior válido. Na segunda maneira, nós fornecemos o custo ótimo acrescido de 0,01 como limite superior.

4.1. Geração das instâncias

Nós seguimos Buzzega e Novellani [2023] e também usamos as instâncias propostas por Jiang et al. [2020], que foram geradas através da inclusão de vértices lockers nos grafos de 40 instâncias benbchmark do problema do caixeiro viajante com janelas de tempo. Em particular, as instâncias originais são dividas em dois conjuntos: o primeiro possui janelas de tempo de amplitude 20 e foram inicialmente propostas por Dumas et al. [1995], enquanto o segundo tem janelas de tempo de amplitude 100 e foram propostas por Gendreau et al. [1998] baseadas nas instâncias propostas por Dumas et al. [1995]. Cada um destes conjuntos pode ser divididos em 4 grupos, cada um com 10 instâncias, com o mesmo número de clientes $n = 20, 40, 60, 100$, cada um destes possuindo uma demanda unitária ($d_i = 1, \forall i \in V_C$). O número de lockers adicionados é $m = n/10$. Cada instância é modificada através da inclusão de m lockers nos m primeiros vértices do seguinte conjunto de coordenadas: (25,0, 25,0), (12,5, 12,5), (37,5, 37,5), (12,5, 25,0), (37,5, 12,5), (12,5, 25,0), (25,0, 37,5), (37,5, 25,0), (25,0, 12,5), e (0, 0). O custo c_e e o tempo de travessia t_e de cada aresta $e \in E$ do grafo são dados pela distância euclideana entre os pontos arrendondada para a segunda casa decimal. O custo c'_{jl} de atribuir um cliente $j \in V_C$ a um locker $l \in V_L$ é dado por $c'_{jl} = 0.5 \times c_{(j,l)}$. Para cada instância, para cada locker $l \in V_L$, tem-se $R_l = 20$ e $\nu_l = 5$. As janelas de tempo do depósito e dos clientes são as mesmas da instância original enquanto que, para cada locker $l \in V_L$, tem-se $[a_l, b_l] = [a_0, b_0]$. A capacidade de cada veículo é dada por $Q = \frac{n}{2}$.

4.2. Comparação com a literatura

Os métodos exatos propostos por Buzzega e Novellani [2023] foram testados nas instâncias descritas na Seção 4.1 considerando um tempo limite de 7200s. Para uma comparação justa, nós comparamos através do site `https://www.cpubenchmark.net/` a versão *single thread* dos processadores utilizados neste trabalho com a do processador usado pela literatura e concluímos que o tempo limite a ser usado neste trabalho é de 5034s.

A Tabela 1 mostra uma comparação com a literatura para instâncias com até 60 clientes. As instâncias estão agrupadas de acordo com a variante (PRVLJT) e de acordo com o número de clientes. Os seguintes cabeçalhos são usados para as colunas. As colunas *Variante* e n indicam respectivamente a variante e o número de clientes associados ao grupo de instâncias correspondente. A Coluna # *Inst* indica o número de instâncias no grupo. A Coluna *Tempo Médio(s)* indica o tempo médio de resolução em segundos considerando todas as instâncias do grupo. Para as instâncias não

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

resolvidas, o tempo limite é usado para o cálculo da média. A Coluna *#Opt* representa o número de soluções comprovadamente ótimas obtidas dentro do limite de tempo. A Coluna *Literatura* apresenta os resultados obtidos pelo melhor método exato proposto por Buzzega e Novellani [2023] enquanto que as Colunas *Este trabalho Sem UB* e *Este trabalho Com UB* apresenta estatística obtidas pelo modelo proposto sem e com o uso do limite superior.

O uso da solução ótima como *upper bound* é interessante para analisarmos o tempo computacional do modelo para resolver o problema provando a otimalidade caso uma heurística que atingiu a melhor solução conhecida fosse utilizada. De toda forma, observando a Tabela 1, nota-se que o método proposto supera o melhor método da literatura para ambas variantes do problema mesmo quando um limite superior não é utilizado.

					Este Trabalho			
			Literatura		Sem UB		Com UB	
Variante	n	#Inst	Tempo Médio(s)	#Opt	Tempo Médio(s)	#Opt	Tempo Médio(s)	#Opt
PRVLJT	20	10	2,5	10	0,0	10	0,0	10
	40	10	1283,2	9	80,1	10	11,2	10
	60	10	5088,2	3	2529,2	8	1716,4	9
PRVL	20	10	2,0	10	2,7	10	0,3	10
	40	10	4639,4	6	144,0	10	43,0	10
	60	10	6187,9	2	3557,7	7	1532,0	9

Tabela 1: Comparação com a literatura

4.3. Resultados detalhados

A Tabela 2 apresenta, para cada experimento considerando o limite superior, o nome da instância correspondente (Coluna *Instância*) o custo ótimo (Coluna $ub*$), o gap percentual da solução ótima em relação ao limite inferior do nó raiz (Coluna Gap_0), e o tempo total em segundos (Coluna *T(S)*). Os nomes de cada instância tem o formato "nXtwYlZ_id", onde X representao o número de clientes, Y representa a amplitude das janelas de tempo (20 ou 100), Z representa o número de lockers e id representa o índice da instância, que varia de 1 a 5.

Observando a Tabela 2, vemos que o método proposto resolveu quase todas as instâncias em razoáveis tempos computacionais. A principal foi a instância n60tw100l6_2, que precisou de 16835s e 11360s para ter a comprovação da otimalidade nas variantes PRVLJT e PRVL, respectivamente. Tais instâncias estão associadas aos maiores gaps na raiz obtidos pelo método.

5. Conclusões e trabalhos futuros

Neste artigo, propusemos um novo método exato para a um problema de roteamento de veículos com lockers com duas variantes: com e sem janelas de tempo. O método proposto é um algoritmo BCP desenvolvido com auxílio do framework VRPSolver.

Aplicamos o algoritmo em 60 instâncias benchmark da literatura com até 60 clientes. Os resultados mostram que o algoritmo proposto supera o melhor método exato da literatura, sendo capaz de resolver a maioria das instâncias em razoáveis tempos computacionais. Várias instâncias foram resolvidas pela primeira vez. Apesar dos bons resultados, algumas instâncias com 60 clientes apresentaram tempos computacionais e *gaps* na raiz relativamente altos.

Como trabalhos futuros, pretendemos estudar o poliedro do problema para identificar cortes capazes de melhorar ainda mais os tempos computacionais obtidos, bem como permitir que o método seja testado em instâncias maiores. Além disso, pretende-se estender o modelo proposto

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

	PRVLJT			PRVL		
Instância	ub^*	Gap_0	T(s)	ub^*	Gap_0	T(s)
n20tw100l2_1	227,465	0,7	0	211,725	0,1	0
n20tw100l2_2	213,245	0,1	0	208,110	0,2	0
n20tw100l2_3	277,805	0,0	0	226,280	0,0	0
n20tw100l2_4	258,295	0,1	0	235,510	0,0	0
n20tw100l2_5	266,490	1,1	0	239,240	1,4	0
n20tw20l2_1	254,620	2,3	0	225,950	2,0	0
n20tw20l2_2	229,695	0,2	0	206,300	0,2	0
n20tw20l2_3	279,575	1,3	0	251,190	2,5	2
n20tw20l2_4	234,070	0,0	0	202,360	0,0	0
n20tw20l2_5	266,815	0,5	0	217,380	0,0	0
n40tw100l4_1	320,790	2,8	10	283,160	3,3	48
n40tw100l4_2	305,700	2,9	39	271,930	0,2	0
n40tw100l4_3	280,515	1,8	8	261,365	2,8	114
n40tw100l4_4	282,210	0,3	0	262,585	1,2	14
n40tw100l4_5	288,090	1,7	6	243,995	0,0	0
n40tw20l4_1	385,930	0,2	0	307,995	2,1	11
n40tw20l4_2	334,090	1,0	1	280,130	1,7	6
n40tw20l4_3	315,025	2,4	11	282,345	4,0	95
n40tw20l4_4	281,990	0,6	1	250,770	1,0	4
n40tw20l4_5	355,765	1,2	2	276,110	2,0	9
n60tw100l6_1	351,000	3,3	3103	299,940	1,0	115
n60tw100l6_2	379,795	4,9	16835	328,855	4,8	11360
n60tw100l6_3	350,710	2,8	2939	301,160	3,0	53
n60tw100l6_4	381,250	2,9	480	327,920	2,6	759
n60tw100l6_5	356,960	2,4	119	307,525	2,6	296
n60tw20l6_1	366,475	1,3	11	317,660	3,0	280
n60tw20l6_2	429,200	1,4	90	345,230	2,7	2077
n60tw20l6_3	400,770	1,7	133	351,055	3,0	926
n60tw20l6_4	405,895	0,1	10	336,140	1,2	180
n60tw20l6_5	405,865	1,8	84	335,710	2,7	993

Tabela 2: Resultados por instância do modelo proposto quando se é fornecido o limite superior válido.

para outras variantes de problemas de roteamentos de veículos que considerem a atribuição de diversos clientes a um único local de atendimento, como é o caso da variante estudada por Tilk et al. [2021].

Referências

Boschetti, M. A. e Novellani, S. (2024). Last-mile delivery with drone and lockers. *Networks*, 83 (2):213–235.

Buzzega, G. e Novellani, S. (2023). Last mile deliveries with lockers: Formulations and algorithms. *Soft Computing*, 27(18):12843–12861.

Costa, L., Contardo, C., e Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4):946–985.

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br

Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1): 80–91.

Dell'Amico, M., Montemanni, R., e Novellani, S. (2023). Pickup and delivery with lockers. *Transportation Research Part C: Emerging Technologies*, 148:104022.

Dumas, Y., Desrosiers, J., Gelinas, E., e Solomon, M. M. (1995). An optimal algorithm for the traveling salesman problem with time windows. *Operations research*, 43(2):367–371.

Gendreau, M., Hertz, A., Laporte, G., e Stan, M. (1998). A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, 46(3):330–335.

Grabenschweiger, J., Doerner, K. F., Hartl, R. F., e Savelsbergh, M. W. (2021). The vehicle routing problem with heterogeneous locker boxes. *Central European Journal of Operations Research*, 29:113–142.

Jiang, L., Dhiaf, M., Dong, J., Liang, C., e Zhao, S. (2020). A traveling salesman problem with time windows for the last mile delivery in online shopping. *International Journal of Production Research*, 58(16):5077–5088.

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.

Praxedes, R., Bulhões, T., Subramanian, A., e Uchoa, E. (2024). A unified exact approach for a broad class of vehicle routing problems with simultaneous pickup and delivery. *Computers & Operations Research*, 162:106467.

Roboredo, M., Sadykov, R., e Uchoa, E. (2023). Solving vehicle routing problems with intermediate stops using vrpsolver models. *Networks*, 81(3):399–416.

Soares, V. C. e Roboredo, M. (2023). On the exact solution of the multi-depot open vehicle routing problem. *Optimization Letters*, p. 1–17.

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

Vincent, F. Y., Susanto, H., Jodiawan, P., Ho, T.-W., Lin, S.-W., e Huang, Y.-T. (2022). A simulated annealing algorithm for the vehicle routing problem with parcel lockers. *IEEE Access*, 10:20764–20782.

DOI: 10.59254/sbpo-2024-193811
https://proceedings.science/p/193811?lang=pt-br