A Branch-and-cut Algorithm for the Multi-depot Inventory-routing Problem with Split Deliveries

Cleder M. Schenekemberg, Cassius T. Scarpin, José E. Pécora Jr., Thiago A. Guimarães

***Abstract*—In this paper, we model and solve a multi-depot inventory-routing problem with split-deliveries (MDIRPSD). The problem emerges under the vendor-managed inventory paradigm when deliveries are made from multiple depots, and the customers can be served several times in the same period. The MDIRPSD integrates supply and distribution decisions and has recently been proposed in the literature. We present a new mathematical formulation, taking into account inventory holding costs, and we also introduce a new inventory policy. We have designed and implemented a new branch-and-cut that overcome a similar approach from the literature. Finally, we report comparative results between the two inventory policies considered.**

***Index Terms*—Vendor-managed inventory, multi-depot, split deliveries, branch-and-cut.**

I. Introdução

Nas últimas décadas, a escalada competitiva global e a consequente redução da margem de lucro vem ensejando maior coordenação de processos e atores no âmbito da cadeia de suprimentos. Ao mesmo tempo, o advento da indústria 4.0 e a disponibilização de dados em tempo real, possibilitam uma agenda cada vez mais colaborativa nas atividades logísticas, com especial ênfase no elo fornecedor-cliente. Neste sentido, paradigmas que requerem a troca de informações e que esbarravam nos obstáculos tecnológicos de anos atrás, assumem um protagonismo cada vez mais transversal. Não apenas pela redução nos custos dos sistemas de troca eletrônica de dados, mas também na minimização de erros e incertezas que esses sistemas possibilitam, em comparação com técnicas indiretas de quantificação das demandas.

Sistemas *Vendor Managed Inventory* (*VMI*) ou estoque gerenciado pelo fornecedor, configuram uma prática gerencial colaborativa com diversos relatos de sucesso em diferentes segmentos industriais (ver [1]). Sob um sistema *VMI*, o fornecedor controla o estoque do cliente, decidindo quando atendê-lo e quanto entregar por ocasião de um serviço. A dinâmica dos estoques é diretamente dependente da demanda do cliente, que pode ser estimada por métodos de previsão, ou observada de forma precisa, através dos supracitados sistemas de troca eletrônica de dados. Segundo [2], a prática *VMI* é mutuamente benéfica, pois os clientes não precisam dispender recursos para controlar seus estoques e emitir pedidos de ressuprimento, enquanto os fornecedores ganham pela maior coordenação das atividades logísticas, particularmente na composição dos roteiros de entrega.

Pelo ângulo operacional, a implementação de um sistema *VMI* implica na resolução de um problema de otimização combinatorial complexo, da classe $\mathcal{NP}-$difícil, denominado *Inventory-Routing Problem* (*IRP*) ou Problema de Roteirização e Estoques. O *IRP* integra, em um mesmo arcabouço, o problema de gerenciamento de estoques e o problema de roteamento de veículos com múltiplos períodos [3]. Nesta abordagem conjunta, compete ao fornecedor decidir: quando servir um cliente, quanto entregar a cada visita realizada e como definir as rotas de entrega. Tais decisões devem garantir que o cliente tenha sua demanda plenamente atendida, sejam pelos estoques mantidos e/ou pelas quantidades recebidas.

Embora a literatura sobre o *IRP* reporte diversas variantes, a estrutura da cadeia de suprimentos predominante é bastante simplificada, envolvendo apenas um fornecedor, que opera a partir de um único depósito. Adicionalmente, proíbe-se que os clientes sejam atendidos com entregas fracionadas, ou seja, o fornecedor programa no máximo um atendimento em um período para cada cliente [4]. Todavia, em um contexto mais realístico, os clientes estão dispersos em uma ampla área urbana e com elevada densidade de tráfego. Ademais, o fornecedor geralmente dispõe de uma rede de depósitos e uma frota de veículos, o que eleva consideravelmente o número de possibilidades de atendimento. Por esses aspectos, o fracionamento das entregas se adere ao melhor uso dos recursos físicos do fornecedor (frota de veículos e múltiplos depósitos), ao mesmo tempo em que gerencia com mais eficiência as questões de tráfego e de logística urbana. Quando um cliente precisa de um reabastecimento em uma área de alto tráfego, o principal objetivo do fornecedor é atender, ainda que parcialmente, a solicitação, usando veículos que visitam o cliente de diferentes depósitos [5]. Essas características definem um *IRP* com múltiplos depósitos e entregas fracionadas, a qual denominamos de *multi-depot inventory-routing problem with split-deliveries* (*MDIRPSD*), e que recentemente foi proposta por [5]. Neste estudo, os autores introduzem o *MDIRPSD* e desenvolvem um algoritmo exato do tipo *branch-and-cut*, além de uma heurística baseada em programação matemática para resolver instâncias de maior porte. Contudo, o problema abordado não contempla o custo de estoque nos clientes, prática muito comum no âmbito do *IRP*, (ver [6], [7], [8], [9]).

Cleder M. Schenekemberg, Cassius T. Scarpin, José E. Pécora Jr. e Thiago A. Guimarães são pesquisadores do Grupo de Tecnologia Aplicada à Otimização (GTAO) da Universidade Federal do Paraná.

José E. Pécora Jr. é membro do Centre Interuniversitaire de Recherche sur les Reseaux d'Entreprise, la Logistique et le Transport (CIRRELT), Canada.

Thiago A. Guimarães é professor do Instituto Federal do Paraná (IFPR).

Tradicionalmente, o fornecedor é livre para escolher a quantidade entregue a cada cliente, limitando-se apenas à capacidade de estocagem e de carregamento do veículo. Essa estratégia define a política de estoque do tipo *maximum level* (*ML*). Alternativamente, a política *order-up-to level* (*OU*) exige que, por ocasião de um atendimento, a quantidade entregue ao cliente complete sua respectiva capacidade de estoque, o que tende a reduzir o número de visitas ao longo do horizonte de planejamento. A política *OU* foi proposta por [10] para o *IRP* clássico e é frequentemente abordada por estudos na área [4]. No âmbito do *IRP* com múltiplos depósitos, esta política foi introduzida por [9], e no melhor de nosso conhecimento, a literatura não reporta trabalhos que a consideram na variante com entregas fracionadas.

Neste artigo, ampliamos o escopo do *MDIRPSD* e reformulamos o problema a fim de incorporar o custo de estoque no processo de otimização. Além disso, introduzimos a política de estoque *order-up to level* no contexto do *MDIRPSD*. Por fim, propomos um novo algoritmo *branch-and-cut* (*B&C*), desenvolvido a partir da nova formulação matemática, e resolvemos o *MDIRPSD* de forma exata, considerando as políticas de estoque *ML* e *OU*. Para a versão simplificada do *MDIRPSD*, com política *ML* e sem custo de estoque, conduzimos uma análise comparativa entre o *B&C* proposto com algoritmo exato apresentado por [5]. Já para a extensão do *MDIRPSD* com custos de estoque, comparamos também as políticas *ML* e *OU* com relação ao custo logístico total.

O restante do artigo está estruturado como segue. Na Seção II definimos o *MDIRPSD* com custos de estoque e apresentamos uma formulação para a política *OU* com entregas fracionadas. Na Seção III apresentamos o algoritmo *branch-and-cut* proposto, enquanto a Seção IV reporta os resultados dos experimentos computacionais realizados. A Seção V tece as conclusões do trabalho.

II. Definição do Problema e Modelagem Matemática

Definimos o *MDIRPSD* sobre um grafo $G = (\mathcal{V}, \mathcal{E})$ onde o conjunto de vértices $\mathcal{V}$, representa a união do conjunto de plantas (ou depósitos) $\mathcal{P}$ e clientes $\mathcal{C}$, enquanto o conjunto $\mathcal{E}$ define as arestas do grafo. Um custo simétrico e não-negativo de transporte $c_{uv} = c_{vu}$ está associado a cada aresta $(u, v) \in \mathcal{E}$, onde $\mathcal{E} = \{(u, v) : u, v \in \mathcal{V} \wedge u, v \oplus \mathcal{P}; u < v\}$, ou seja, não há arcos entre as plantas, representado pelo operador "ou-exclusivo ($\oplus$)".

O conjunto $\mathcal{T} = \{1, ..., p\}$ define um horizonte de planejamento com p períodos. Uma frota homogênea e limitada de veículos é definida pelo conjunto $\mathcal{K}$. Cada veículo $k \in \mathcal{K}$ possui a mesma capacidade Q, e pode ser designado para uma única planta $j \in \mathcal{P}$ em cada período $t \in \mathcal{T}$. Assume-se que a capacidade de produção e estoque em cada planta j é ilimitada.

Cada cliente $l \in \mathcal{C}$ tem uma demanda determinística não nula em cada período, dado por d_l^t, onde $d_l^0 = 0$. A capacidade máxima de estoque é U_l, sendo constante ao longo de $\mathcal{T}$. Cada cliente l incorre em um custo de estoque h_l por unidade estocada por período. O nível de estoque ao final de cada período $t \in \mathcal{T} \cup \{p + 1\}$ é dado por I_l^t e este não pode ser negativo. Dessa maneira, efeitos das decisões tomadas no último período p, e seu respectivo impacto no período imediatamente seguinte são considerados na formulação, $(p + 1)$. No início do horizonte de planejamento, o cliente pode ter uma certa disponibilidade de estoque, conhecida *a priori*, definida por I_l^0.

Dois tipos de entregas fracionadas são permitidas a um cliente l no mesmo período t: quando dois ou mais veículos de uma mesma planta j atendem ao referido cliente, ou quando este é servido por dois ou mais veículos de plantas distintas. No entanto, toda rota deve iniciar e terminar na mesma planta de origem.

O objetivo do *MDIRPSD* é minimizar o custo logístico total, dado pela soma dos custos de estoque e de transporte, determinando:

- quando visitar, quanto entregar e a partir de quais plantas cada cliente deve ser atendido;
- quantos veículos utilizar em cada planta em cada período;
- como combinar as entregas dos clientes nas rotas dos veículos;

Com relação às variáveis de decisão, cada planta j deve determinar a quantidade q_{jl}^{kt} entregue ao cliente l, com o veículo k no período t. Pelo *timing* das atividades considerado em [5], destacamos que a quantidade entregue ao cliente em t só poderá ser utilizada em $t + 1$. De forma equivalente, as entregas são realizadas após a ocorrência da demanda, o que, em um contexto prático, acontece após o cliente encerrar as atividades no período t. As variáveis restantes são definidas a seguir.

- $Y_{jl}^{kt} = 1$ se o veículo k da planta j visita o vértice l no período t, onde $l \in \mathcal{V}$, 0 caso contrário;
- $y_{uv}^{kjt} = 1$ se o veículo k da planta j viaja do cliente u ao cliente v no período t, 0 caso contrário.
- $y_{jv}^{kjt} \in \{0, 1, 2\}$ assume o valor 1 se o veículo k da planta j viaja da referida planta ao cliente l no período t. Quando $y_{jv}^{kjt} = 2$, uma entrega direta é definida, 0 caso contrário.

Utilizamos a variável $Y_{jj}^{kt} = 1$ para controlar a saída do veículo k da planta j no período t. Finalmente, o *MDIRPSD* é formulado por (1)−(14).

$$\min \sum_{t \in \mathcal{T} \cup \{p+1\}} \sum_{l \in \mathcal{C}} h_l I_l^t + \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} \sum_{(u,v) \in \mathcal{E}} c_{uv} y_{uv}^{kjt} \quad (1)$$

sujeito à

$$I_l^t = I_l^{t-1} + \sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} q_{jl}^{k,t-1} - d_l^t \quad l \in \mathcal{C}, t \in \mathcal{T} \cup \{p+1\} \quad (2)$$

$$I_l^t \leq U_l \quad l \in \mathcal{C}, t = p+1 \quad (3)$$

$$I_l^t + \sum_{j \in \mathcal{P}} \sum_{k \in \mathcal{K}} q_{jl}^{kt} \leq U_l \quad l \in \mathcal{C}, t \in \mathcal{T} \quad (4)$$

$$q_{jl}^{kt} \leq U_l Y_{jl}^{kt} \quad l \in \mathcal{C}, j \in \mathcal{P}, k \in \mathcal{K}, t \in \mathcal{T} \quad (5)$$

$$\sum_{l \in \mathcal{C}} q_{jl}^{kt} \leq Q Y_{jj}^{kt} \quad j \in \mathcal{P}, k \in \mathcal{K}, t \in \mathcal{T} \quad (6)$$

$$\sum_{j \in \mathcal{P}} Y_{jj}^{kt} \leq 1 \quad k \in \mathcal{K}, t \in \mathcal{T} \quad (7)$$

$$\sum_{\substack{u \in \mathcal{V} \\ l < u}} y_{ul}^{kjt} + \sum_{\substack{u \in \mathcal{V} \\ u < l}} y_{lu}^{kjt} = 2 Y_{jl}^{kt} \quad l \in \mathcal{V}, j \in \mathcal{P}, k \in \mathcal{K}, t \in \mathcal{T} \quad (8)$$

$$\sum_{l\in S}\sum_{\substack{u\in S\\ l<u}} y_{lu}^{kjt} \leq \sum_{l\in S} Y_{jl}^{kt} - Y_{jm}^{kt} \qquad S \subseteq \mathcal{C}, |S| \geq 2, m \in S, j \in \mathcal{P}, k \in \mathcal{K}, t \in \mathcal{T} \tag{9}$$

$$I_l^t \in \mathbb{R}^+ \qquad l \in \mathcal{C}, t \in \mathcal{T} \cup \{p+1\} \tag{10}$$

$$q_{jl}^{kt} \in \mathbb{R}^+ \qquad j \in \mathcal{P}, l \in \mathcal{C}, k \in \mathcal{K}, t \in \mathcal{T} \tag{11}$$

$$Y_{jl}^{kt} \in \{0,1\} \qquad j \in \mathcal{P}, l \in \mathcal{V}, k \in \mathcal{K}, t \in \mathcal{T} \tag{12}$$

$$y_{jv}^{kjt} \in \{0,1,2\} \qquad v \in \mathcal{C}, j \in \mathcal{P}, k \in \mathcal{K}, t \in \mathcal{T} \tag{13}$$

$$y_{uv}^{kjt} \in \{0,1\} \qquad u, v \in \mathcal{C}, j \in \mathcal{P}, k \in \mathcal{K}, t \in \mathcal{T} \tag{14}$$

A função objetivo (1) computa os custos totais, dado pela soma dos custos de estoque e transporte. A função objetivo deve ser minimizada. As restrições (2) garantem a conservação de fluxo de estoque, enquanto (3) limitam os níveis de estoques nos clientes. A política *ML* considerando o *timing* das atividades é formulada pelas restrições (4). Já as restrições (5) conectam a quantidade entregue pela planta j ao cliente l com o veículo k, caso o cliente seja visitado pelo referido veículo da referida planta. As restrições (6) impedem que a capacidade de um veículo k seja excedida ao atender um conjunto de clientes, enquanto (7) garantem que cada veículo k seja designado a uma única planta j em cada período t. As restrições (8) e (9) estabelecem a conectividade dos vértices e proíbem a formação de sub rotas, respectivamente. Particularmente, as restrições (9) impedem a ocorrência de subciclos, impondo para cada subconjunto próprio e não vazio S de $\mathcal{C}$, que o número total de arestas entre os nós de S percorridas pelo veículo k da planta j deve ser no máximo $|S|-1$. O domínio das variáveis de decisão é dado pelas restrições (10)−(14). As entregas fracionadas são permitidas pela não inclusão de uma restrição de unicidade de atendimento, que pode ser formulada como $\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} Y_{jl}^{kt} \leq 1, l \in \mathcal{C}, t \in \mathcal{T}$.

Destacamos que o modelo matemático (1)−(14) é suficientemente flexível para representar o problema proposto por [5]. Para isso, basta definir $h_l = 0$, $l \in \mathcal{C}$. Adicionalmente, albergamos a quebra de simetria nas variáveis binárias que representam as decisões de rota via entrega direta, ao modelar $y_{jv}^{kjt} = \{0,1,2\}$, limitando-as a um único arco. Tal formulação simplifica o modelo proposto por [5], reduzindo a cardinalidade das arestas do conjunto $\mathcal{E}$, o que melhora o desempenho dos métodos exatos na resolução de problemas de roteirização [7].

Em nosso melhor conhecimento, a literatura do *IRP* não reporta nenhuma formulação para a política *OU* quando as entregas aos clientes podem ser fracionadas. De maneira a cobrir essa lacuna, definimos uma variável binária adicional Z_l^t, que assume o valor 1 quando o cliente l é visitado ao menos uma vez no período t, e 0 caso contrário. As restrições (15)−(17) asseguram a política *OU* para o *MDIRPSD*. A constante $M = |\mathcal{P}||\mathcal{K}|$ é um valor suficientemente grande.

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} Y_{jl}^{kt} \leq M Z_l^t \qquad l \in \mathcal{C}, t \in \mathcal{T} \tag{15}$$

$$I_l^t + \sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}} q_{jl}^{kt} \geq U_l Z_l^t \qquad l \in \mathcal{C}, t \in \mathcal{T} \tag{16}$$

$$Z_l^t \in \{0,1\} \qquad l \in \mathcal{C}, t \in \mathcal{T} \tag{17}$$

De maneira a fortalecer os limitantes duais, nós apresentamos um conjunto de desigualdades válidas, propostas por [11] e [12], para a variante clássica do *IRP*.

$$y_{jl}^{kjt} \leq 2Y_{jl}^{kt} \qquad j \in \mathcal{P}, l \in \mathcal{C}, k \in \mathcal{K}, t \in \mathcal{T} \tag{18}$$

$$y_{lu}^{kjt} \leq Y_{jl}^{kt} \qquad j \in \mathcal{P}, l, u \in \mathcal{C}, l < u, k \in \mathcal{K}, t \in \mathcal{T} \tag{19}$$

$$y_{lu}^{kjt} \leq Y_{ju}^{kt} \qquad j \in \mathcal{P}, l, u \in \mathcal{C}, l < u, k \in \mathcal{K}, t \in \mathcal{T} \tag{20}$$

$$Y_{jl}^{kt} \leq Y_{jj}^{kt} \qquad j \in \mathcal{P}, l \in \mathcal{C}, k \in \mathcal{K}, t \in \mathcal{T} \tag{21}$$

As desigualdades (18) elevam os limitantes inferiores do modelo, quando um cliente l é atendido por uma entrega direta pelo veículo k designado à planta j no período t. Analogamente, (19) e (20) representam o caso de múltiplos clientes servidos em uma mesma rota. Já as desigualdades (21) asseguram que o cliente l poderá ser atendido pelo veículo k que parte da planta j no período t, somente se esse veículo for utilizado.

Em (22), apresentamos uma adaptação para o *MDIRPSD* das restrições de quebra de simetria, introduzidas por [12] para o *IRP* básico. Tais conjuntos asseguram que um veículo de índice k só será utilizado por uma planta j, quando o veículo de índice $k-1$ já estiver em uso por alguma planta.

$$\sum_{j\in\mathcal{P}} Y_{jj}^{kt} \leq \sum_{j\in\mathcal{P}} Y_{jj}^{k-1,t} \qquad t \in \mathcal{T}, k \in \mathcal{K}, k > 1 \tag{22}$$

Por fim, consideramos em (23) as desigualdades propostas por [9]. O lado direito calcula a razão entre necessidade mínima de entrega do cliente no intervalo $[t_1, t_2]$, pelo menor valor entre as capacidades do veículo e de estocagem do cliente. O teto dessa divisão indica o menor número de entregas necessárias para atender à demanda acumulada no referido intervalo.

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}\sum_{t=t_1}^{t_2} Y_{jl}^{kt} \geq \left\lceil \frac{\sum_{t=t_1}^{t_2} d_l^t - U_l}{min\{Q, U_l\}} \right\rceil \qquad l \in \mathcal{C}, t_1, t_2 \in \mathcal{T}, t_2 > t_1 \tag{23}$$

III. Algoritmo Branch-and-Cut

Por conta da sua complexidade combinatorial, o número de subconjuntos necessários para gerar todas as restrições de eliminação de subrotas (RES) em (9) é demasiadamente grande, fazendo com que o processo enumerativo pleno seja impraticável. Para superar essa limitação, tais restrições podem ser geradas e adicionadas no decorrer do processo de otimização. Nós delineamos uma abordagem exata para resolver o modelo apresentado na Seção II, onde as RES são dinamicamente adicionadas à arvore de busca, sempre que sub rotas forem identificadas na solução corrente. Esta técnica é conhecida como *branch-and-cut* (*B&C*), e combina a clássica metodologia *branch-and-bound* (*B&B*) com algoritmos de planos de corte.

Sempre que existirem variáveis de domínio inteiro com valores fracionários na solução corrente, uma ramificação deve ser imposta. Após uma criteriosa seleção, a variável fracionária escolhida $x = a$, limita-se à $x \geq \lceil a \rceil$ ou $x \leq \lfloor a \rfloor$. Para o *MDIRPSD*, priorizamos a ramificação das variáveis de saída de um veículo da planta Y_{jj}^{kt}, seguido das variáveis de atendimento de cliente Y_{jl}^{kt} e por último, as variáveis de rota y_{uv}^{kjt}.

No início do processo de busca, todas as restrições, exceto o conjunto (9), e todas as desigualdades válidas são geradas e adicionadas ao nó raiz da árvore do *B&B*. A cada nó resolvido pelo método *B&B*, um algoritmo de rastreio por RES violadas é então aplicado. Para essa finalidade nós utilizamos o pacote *CVRPSEP*, desenvolvido por [13] para o *capacitated vehicle routing problem (CVRP)* ou problema de roteamento de veículos capacitados. Uma vez que o MDIRPSD generaliza o *CVRP*, o CVRPSEP atua sobre a estrutura de roteirização do problema. A cada nó resolvido na árvore B&B, fornecemos ao CVRPSEP as variáveis de roteirização associadas à cada veículo, de cada planta e em cada período. O pacote identifica as RES violadas, e adiciona, em tempo de processamento, um conjunto de cortes de capacidade. Neste ponto, um novo subproblema é gerado pela ramificação de uma nova variável fracionária, e o problema original é então reotimizado a partir de um novo nó da árvore *B&B*. O processo é finalizado após um limite de tempo estabelecido ou após o término da exploração dos nós gerados pela árvore.

Cabe destacar que utilizamos o *framework* do B&C já disponível no *solver* Gurobi, que gerencia toda a estrutura da árvore de busca. Baseado na topologia conhecida do MDIRPSD, a ramificação de variáveis é feita pelo parâmetro *BranchPriority* do solver. Durante o processo de busca, verificamos se a solução corrente de cada nó otimizado é fracionária ou inteira. No primeiro caso, o CVRPSEP identifica as RES violadas e insere um corte válido ao modelo. Caso a solução seja inteira, as RES violadas são adicionadas como *lazy constraints* pelo framework do Gurobi.

Pela complexidade das RES, o algoritmo *B&C* muitas vezes não é capaz de encontrar soluções incumbentes, especialmente para problemas de médio e grande porte. Para contornar essa limitação, propomos uma estratégia para a obtenção de uma solução inicial factível, partindo de uma reformulação do modelo da Seção II. Para tanto, desconsideramos as variáveis de rota y_{uv}^{kjt}. A nova função objetivo, dada pela equação (24), considera o custo de entrega direta, caso o cliente seja servido pelo veículo k, da planta j no período t.

$$\min \sum_{t \in \mathcal{T} \cup \{p+1\}} \sum_{l \in \mathcal{C}} h_l I_l^t + \sum_{j \in \mathcal{P}} \sum_{l \in \mathcal{C}} \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} c_{jl} Y_{jl}^{kt} \qquad (24)$$

A equação (24) está sujeita às restrições (2)−(7) e também (10)−(12), resultando em um modelo de fluxo com entregas diretas. A política *OU* é igualmente contemplada adicionando as restrições (15)−(17) à este modelo de fluxo.

Para gerar as rotas iniciais e completar a solução incumbente do *MDIRPSD*, resolvemos os problemas de caixeiro viajante, ou *traveling salesman problem* (*TSPs*), para as triplas (j, k, t) associadas às variáveis $Y_{jl}^{kt} = 1$ pelo método proposto por [14]. A solução inicial obtida é então inserida no nó raiz do *B&C* proposto. Um esquema para o algoritmo *B&C* é apresentado pelo **Algoritmo 1**.

Algorithm 1 Pseudo código para o algoritmo *B&C*

```
Obter uma solução inicial e adicionar ao nó raiz da árvore de busca.
No nó raiz da árvore de busca, gerar (2)–(8), (10)–(14) e todas as desigualdades válidas (18)–(23).
se Política de Estoque for OU então
    Adicionar as restrições (15)−(17)
fim se
Solução do Subproblema: resolver o problema de programação linear (PL) associado ao nó.
Critério de Parada:
se Não existirem mais nós para serem avaliados então
    Pare.
senão
    Selecione um novo nó da árvore B&B.
fim se
enquanto A solução do PL conter sub rotas faça
    Adicione as RES associadas.
    Solução do Subproblema: resolver o PL associado ao nó.
fim enquanto
se A solução do PL é inteira então
    Ir para o Critério de Parada.
senão
    Ramificação: Escolher e ramificar uma variável fracionária.
    Ir para o Critério de Parada.
fim se
```

O *B&C* proposto nesse estudo destaca-se em relação ao método exato introduzido por [5] pelos seguintes elementos:

- Nós desenvolvemos um método de solução inicial, superando a limitação característica de esquemas *B&C*, como em [5], que muitas vezes são incapazes de prover ao menos uma solução viável em problemas de médio e grande porte.
- Nossa formulação (ver Seção II) é mais compacta, ao considerar variáveis para entregas diretas (y_{jv}^{kjt}) e de controle de saída dos veículos (Y_{jj}^{kt}).
- Utilizamos o pacote CVRPSEP para separação e tratamento das RES, tanto para solução para variáveis inteiras com valores fracionários quanto para soluções inteiras. Já [5] emprega um procedimento heurístico para tal.
- Por fim, nosso *B&C* é flexível o bastante para resolver a política OU, desconsiderada no trabalho de [5].

IV. Experimentos Computacionais

Os testes computacionais foram realizados em processadores Intel(R) Xeon(R) CPU E5-2630 v2, 2.60GHz, com 16 GB de memória RAM e sistema operacional Fedora Linux. O algoritmo $B\&C$, descrito na Seção III, foi implementado em C++ e os modelos de programação linear inteira foram resolvidos com o *solver* Gurobi 8.1.0, processando simultaneamente em até seis *threads*. Cabe destacar que nós modificamos as estruturas de dados e as principais funções do pacote CVRPSEP, dado que o pacote não é *thread safe*, a fim de garantir que os resultados não sejam comprometidos durante a otimização. O algoritmo *B&C* de [5], foi implementado C++, e otimizado

com o solver CPLEX 12.6.1. Os testes foram realizados pelos autores em um processador Intel Core i7-6500U CPU 2.50GHz com 8 GB de RAM e sistema operacional Scientific Linux 6.6 por até 21700 segundos, mesmo tempo utilizado para o nosso B&C nas Tabelas I e II. Para os testes comparativos entre as políticas de estoque, optamos pela parametrização mais frequente na literatura, e o *B&C* foi executado por até 7200 segundos para cada instância.

Nosso *B&C* foi testado sobre um conjunto de 100 instâncias, separadas em classes de 3 e 6 períodos. Os problemas foram adaptados para a variante com múltiplos depósitos por [5], a partir das instâncias do IRP clássico, propostas por [11]. As instâncias estão agrupadas em 5, 10, 15, 20, 25, 30, 35, 40, 45 e 50 clientes, com cinco instâncias por grupo. Todas elas possuem frota com 3 veículos e o número de plantas depende do número de clientes. Pela quantidade de instâncias testadas, optamos por reportar os resultados consolidados na média de cada grupo. Os resultados detalhados neste artigo estão disponíveis sob requisição.

A Tabela I reporta os resultados médios por grupo para a classe com 3 períodos ($|\mathcal{T}| = 3$). As duas primeiras colunas à esquerda informam a quantidade de clientes no grupo e o número de plantas. Da esquerda para a direita, tem-se o número de soluções ótimas encontradas, a média dos *upper bounds* ($\overline{UB}$) e a média dos tempos de processamento. Salientamos que [5] não apresentam os *lower bounds* das instâncias, e atesta-se otimalidade quando o tempo de processamento é inferior ao limite de 21000 segundos. Sequencialmente apresentamos as estatísticas médias do nosso $B\&C$, iniciando pelo número de soluções ótimas encontradas, média dos *upper bounds* ($\overline{UB}$), média dos *lower bounds* ($\overline{LB}$), média percentual dos *gaps* ($\overline{GAP}(\%)$), sendo que, para cada instância, $GAP = \left(\frac{UB-LB}{UB}\right)$, e a média dos tempos de processamento. As duas últimas colunas trazem as análises comparativas médias entre os algoritmos, onde, para cada instância, o *gap* entre os *upper bounds* foi calculado como $GAP = \left(\frac{UB_{nosso}-UB_{bertazzi}}{UB_{bertazzi}}\right)$ e o *gap* entre os tempos de processamento foi obtido por $GAPT(s) = \left(\frac{T(s)_{nosso}-T(s)_{bertazzi}}{T(s)_{bertazzi}}\right)$. Valores negativos nas colunas $\Delta(\%)$ são favoráveis ao nosso algoritmo.

Os resultados mostram que o *B&C* proposto foi capaz de provar a otimalidade em 19 das 20 instâncias até 20 clientes, sendo que o algoritmo *B&C* de [5] provam a otimalidade apenas para instâncias com 5 clientes. De forma geral, nosso método encontrou 28 soluções ótimas contra 10 do método de [5]. Além disso, nosso *B&C* foi capaz de obter soluções ótimas para duas instâncias com 45 clientes, o que endossa a eficiência da formulação matemática proposta. Analisando o desempenho comparativo geral, nosso *B&C* reduziu a média dos *upper bounds* em 16%, demandando pouco mais da metade do tempo de processamento.

A classe de 50 instâncias com 6 períodos apresenta seus resultados na Tabela II, que obedece a mesma sequencia anterior. Pela dificuldade inerente ao problema, foi possível encontrar soluções ótimas em apenas 5 instâncias, 4 delas para o grupo com 5 clientes, enquanto instâncias com 50 clientes apresentam média dos *gaps* superior à 30%. Esse padrão, todavia, é esperado para métodos exatos em problemas dessa natureza [4]. Ainda assim, nosso método foi superior para todos os grupos de instâncias, exceto aquelas com 40, 45 e 50 clientes. Isso pode ser explicado pelo fato do nosso *B&C* partir de uma solução inicial e da estratégia de ramificação empregada, o que acaba reorientando o processo de busca na árvore *B&B*. Ainda assim, nosso método reduziu em até 20% os *upper bounds* para a classe com 20 clientes, e $7,7\%$ na média geral.

O segundo grupo de análise compara a classe de instâncias com 3 períodos entre as políticas de estoque *ML* e *OU*, sem considerar os custos de manutenção, ou seja, $h_l = 0$, $l \in \mathcal{C}$. Da esquerda para a direita, a Tabela III reporta, para cada política, o número de soluções ótimas encontradas, média dos *upper bounds* ($\overline{UB}$), média dos *lower bounds* ($\overline{LB}$), média percentual dos *gaps* ($\overline{GAP}(\%)$) e média dos tempos de processamento. O cálculo dos *gaps* de cada instância é idêntico às Tabelas I e II. As últimas colunas comparam a média dos *UB's* entre as políticas, sendo que o *gap* é calculado como $\Delta UB = \frac{UB_{OU}-UB_{ML}}{UB_{ML}}$. Os *gaps* dos tempos de processamento individuais são obtidos de forma equivalente. Para a classe com 3 períodos, o algoritmo *B&C* encontrou 22 soluções ótimas na política *ML* e 17 na *OU*. Destacamos que os testes foram executados por até 7200 segundos, o que reduziu o número de ótimos encontrados pela política *ML* em relação à Tabela I. Já pelo aspecto do custo total, a política *OU* é em média $10,6\%$ mais custosa, mesmo sem considerar o custo de estocagem. Esse padrão de resposta é condizente com a formulação do problema, pois, em último caso, a política *ML* recai na *OU*. Considerando apenas as soluções ótimas nas duas políticas, o custo total da política *OU* é maior em $5,7\%$ para $|\mathcal{T}| = 3$ e $3,7\%$ para $|\mathcal{T}| = 6$, em relação à política *ML*.

Por fim, apontamos que os resultados para a política *ML* das Tabelas III e IV são equivalentes às Tabelas I e II, diferindo pelo tempo de processamento do nosso algoritmo B&C. Mesmo com 7200 segundos, nosso método foi capaz de provar a otimalidade para 22 instâncias com $|\mathcal{T}| = 3$, contra 10 de [5]. Já para $|\mathcal{T}| = 6$ provamos 2 ótimos adicionais que os referidos autores. Há melhorias também na qualidade das soluções encontradas. Evidenciamos dessa maneira a superioridade do nosso algoritmo, mesmo diante das diferenças computacionais de hardware e software com o método de [5].

Também para os testes sem considerar os custos de estocagem, a complexidade do problema fica evidente na classe de instâncias com 6 períodos, conforme apresentado na Tabela IV. Para o menor grupo, com 5 clientes, o algoritmo $B\&C$ provou a otimalidade para todas as instâncias na política *OU*, contra apenas 2 instâncias na política *ML*. Para os grupos com mais clientes, o menor número de visitas da *OU* não é suficiente para viabilizar a otimização, e o tempo máximo de processamento foi atingido em praticamente todas as instâncias a partir do grupo com 10 clientes. Pelo fato das instâncias com maior número de clientes estarem mais distantes da prova de otimalidade, a comparação entre a política *ML* e *OU* deixa de ser acurada.

Comparamos também as duas políticas para a formulação apresentada na Seção II, considerando os custos de estocagem nos clientes. A Tabela V reporta a mesma estrutura da tabela

TABELA I
COMPARAÇAO COM BERTAZZI [5] PARA $|T| = 3$ PERÍODOS

$\|\mathcal{T}\| = 3$		Bertazzi [5]			B&C Proposto					Comparação	
$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	*OPT*	$\overline{UB}$	$\overline{T(s)}$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	$\Delta(\%)\overline{UB}$	$\Delta(\%)\overline{T(s)}$
5	2	5	1428.0	496.5	5	1428.0	1428.0	0.0	1.0	0.0	-99.4
10	2	3	2137.9	9596.6	5	2124.3	2124.3	0.0	143.0	-0.6	-99.1
15	2	2	3098.8	18531.5	4	2843.9	2843.7	0.0	4339.0	-7.8	-80.0
20	3	0	3979.5	21700.0	5	3572.9	3572.9	0.0	4469.0	-10.0	-79.4
25	4	0	4967.8	21700.0	2	3817.8	3594.1	5.8	16223.0	-22.0	-25.2
30	4	0	4971.0	21700.0	3	3680.2	3567.9	2.8	13435.0	-26.2	-38.1
35	5	0	6362.4	21700.0	2	4723.7	4393.1	6.4	14901.0	-26.0	-31.3
40	5	0	6316.4	21700.0	0	4623.3	3957.7	13.8	21700.0	-27.6	0.0
45	6	0	6514.3	21700.0	2	5319.6	4723.5	9.5	20735.0	-18.8	-4.4
50	6	0	7435.0	21700.0	0	5581.1	4773.2	12.9	21700.0	-25.0	0.0
Média			**4721.1**	**18052.5**		**3771.5**	**3497.8**	**5.1**	**11764.6**	**-16.4**	**-45.7**

TABELA II
COMPARAÇAO COM BERTAZZI [5] PARA $|T| = 6$ PERÍODOS

$\|\mathcal{T}\| = 6$		Bertazzi [5]			B&C Proposto					Comparação	
$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	*OPT*	$\overline{UB}$	$\overline{T(s)}$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	$\Delta(\%)\overline{UB}$	$\Delta(\%)\overline{T(s)}$
5	2	1	3424.2	19097.0	4	3416.0	3367.3	1.5	8555.0	-0.2	-60.6
10	2	0	5429.6	21700.0	0	4827.2	4186.7	13.1	21700.0	-11.2	0.0
15	2	0	9451.2	21700.0	1	8199.4	7566.4	7.6	18233.0	-13.2	-16.0
20	3	0	11569.3	21700.0	0	8716.6	7373.9	14.0	21700.0	-23.6	0.0
25	4	0	10569.0	21700.0	0	8663.6	7877.2	8.5	21700.0	-18.0	0.0
30	4	0	11780.3	21700.0	0	10319.1	8734.7	13.8	21700.0	-12.8	0.0
35	5	0	12523.3	21700.0	0	11357.0	9844.5	12.6	21700.0	-9.2	0.0
40	5	0	12585.0	21700.0	0	13385.8	10279.8	23.2	21700.0	6.4	0.0
45	6	0	11932.1	21700.0	0	12470.6	9064.5	26.9	21700.0	4.2	0.0
50	6	0	13853.4	21700.0	0	13920.1	9288.6	33.2	21700.0	0.6	0.0
Média			**10311.7**	**21439.7**		**9527.5**	**7758.4**	**15.4**	**20038.8**	**-7.7**	**-7.7**

TABELA III
COMPARAÇÃO ENTRE AS POLÍTICAS *ML* E *OU* SEM CUSTO DE ESTOQUE E $|\mathcal{T}| = 3$ PERÍODOS

$\|\mathcal{T}\| = 3$		ML					OU					$(OU - ML)/ML$	
$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	$\Delta(\%)\overline{UB}$	$\Delta(\%)\overline{T(s)}$
5	2	5	1428.0	1428.0	0.0	1.0	5	1464.3	1464.3	0.0	0.4	3.1	-60.0
10	2	5	2124.3	2124.3	0.0	143.0	5	2237.9	2237.9	0.0	135.0	5.6	-5.6
15	2	4	2843.8	2795.1	1.2	1504.0	5	3055.1	3055.0	0.0	1995.0	7.7	32.6
20	3	4	3572.9	3523.0	1.6	1966.0	2	3801.3	3705.5	2.7	4430.0	6.6	125.3
25	4	1	3817.8	3487.2	8.7	6030.0	0	4073.1	3756.4	7.6	7200.0	6.5	19.4
30	4	1	3712.9	3496.4	5.4	6185.0	0	3947.5	3612.4	8.6	7200.0	6.7	16.4
35	5	2	4777.9	4334.5	8.5	6210.0	0	5821.0	4589.0	19.6	7200.0	21.4	15.9
40	5	0	4851.5	3904.0	19.3	7200.0	0	5794.1	4170.5	27.4	7200.0	22.8	0.0
45	6	0	5717.7	4596.8	17.8	7200.0	0	6505.4	4760.8	25.3	7200.0	16.2	0.0
50	6	0	6513.3	4642.7	27.3	7200.0	0	7055.5	4783.9	31.5	7200.0	8.9	0.0
Média			**3936.0**	**3433.2**	**9.0**	**4363.9**		**4375.5**	**3613.6**	**12.3**	**4976.0**	**10.6**	**14.4**

anterior. Com relação ao desempenho do algoritmo *B&C*, verificamos 17 soluções ótimas encontradas para a política *OU* e 24 para a política *ML*. A média dos *gaps* no entanto, ficaram mais distantes entre as políticas. Os custos de estoque são representativos no processo de otimização, notadamente pela compensação de custos entre transporte e estoque, haja vista a média mais justa dos *UBs* e *LBs*, especialmente para os grupos com mais clientes.

Já a análise comparativa destaca ainda mais a relevância dos custos de estocagem. Como a política *OU* mantém níveis de estoque mais elevados, a diferença é mais díspar. Em valores médios para o conjunto das 50 instâncias da classe, a política *OU* é $17,2\%$ mais custosa que a política *ML*. Esse dado é condizente com os resultados reportados por [11], [15], [9],

TABELA IV
COMPARAÇÃO ENTRE AS POLÍTICAS *ML* E *OU* SEM CUSTO DE ESTOQUE E $|T| = 6$ PERÍODOS

$\|\mathcal{T}\| = 6$			ML				OU					$(OU - ML)/ML$	
$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	$\Delta(\%)\overline{UB}$	$\Delta(\%)\overline{T(s)}$
5	2	2	3416.0	3305.4	3.4	4565	5	3704.6	3704.6	0.0	103	6.7	-80.7
10	2	0	4834.5	4119.1	14.7	7200	0	5129.7	4435.7	13.3	7200	6.1	0.0
15	2	1	8202.6	7490.2	8.6	6642	0	8723.1	7440.0	14.6	7200	6.4	12.7
20	3	0	8839.4	7329.1	15.4	7200	0	9641.7	7501.1	20.0	7200	8.4	0.0
25	4	0	9251.7	7807.6	14.3	7200	0	9432.1	7867.2	15.4	7200	2.1	0.0
30	4	0	11068.0	8673.9	20.8	7200	0	11181.4	8629.0	21.3	7200	0.6	0.0
35	5	0	12004.4	9322.8	21.9	7200	0	12335.6	9359.6	23.6	7200	2.5	0.0
40	5	0	13385.9	9793.4	26.9	7200	0	13618.1	10171.6	25.1	7200	1.5	0.0
45	6	0	12470.6	7987.6	35.7	7200	0	12179.7	8840.6	27.0	7200	-2.2	0.0
50	6	0	13920.1	8115.8	41.7	7200	0	13945.1	8645.4	38.0	7200	0.2	0.0
Média			**9739.3**	**7394.5**	**20.3**	**6880.7**		**9989.1**	**7659.5**	**19.8**	**6490.3**	**3.2**	**-6.8**

que resolvem diferentes variações do *IRP* sob a política de estoque *OU*.

Apresentamos na Tabela VI os resultados equivalentes para a classe com 6 períodos, com estrutura idêntica à Tabela V. Os dados ressaltam duas direções. A primeira equipara a dificuldade de resolução do *MDIRPSD* entre as políticas, à medida em que o número de variáveis aumenta com a expansão do número de períodos no horizonte de planejamento. As médias dos *gaps* são equivalentes para todos os grupos, bem como quanto ao tempo de processamento. Ademais, ao considerar custos de estocagem, a discrepância entre as políticas fica mais evidente, e não se verifica mais casos em que a política *OU* possui resultados médios superiores à *ML*. Considerando apenas as soluções ótimas nas duas políticas, o custo total da política *OU* é maior em 11.1% para $|\mathcal{T}| = 3$ e 16.3% para $|\mathcal{T}| = 6$, em relação à política *ML*. Por fim, o efeito da incorporação do custo de estocagem quando $|\mathcal{T}| = 3$ e $|\mathcal{T}| = 6$, pode ser feito comparando em pares as Tabelas III e V, além de IV e VI, respectivamente. A mudança na função objetivo (ver Seção II) impacta no desempenho do algoritmo *B&C*, melhorando a estratégia de ramificação. Os *gaps* foram menores quando o custo de estoque são considerados, tanto para a política *ML* quanto para a *OU*. Esse fato decorre da melhoria nos valores de *UB*, especialmente nas instâncias maiores. Contudo, há maior dificuldade na prova de otimalidade, o que é alcançado com mais frequência nas instâncias de menor porte, sem a imposição do custo de estoque.

V. CONCLUSÕES

Neste trabalho abordamos um problema prático, que emerge do paradigma do estoque gerenciado pelo fornecedor no âmbito da logística urbana, denominado *Multi Depot Inventory Routing Problem with Split Deliveries* (*MDIRPSD*). Assente ao trabalho de [5], que introduziu o problema na literatura, nós expandimos a formulação e incorporamos os custos de estocagem no cliente. Ademais, adaptamos a política de estoque *OU* para o *MDIRPSD* e desenvolvemos um novo algoritmo *B&C* com um mecanismo de obtenção de uma solução inicial, a partir de uma nova formulação do problema. Os resultados computacionais apontam que nosso algoritmo é superior ao $B\&C$ proposto por [5], sendo capaz de reduzir a média dos *UBs* para praticamente todos os grupos de instâncias. Ao mesmo tempo, nosso algoritmo foi capaz de encontrar 18 novas otimalidades para a classe de instâncias com 3 períodos e 4 novas otimalidades para a classe com 6 períodos. A análise comparativa entre as políticas *ML* e *OU* demonstra que níveis mais elevados de estoque geram soluções mais custosas, e destaca a importância de se considerar os custos de estoque no processo decisório.

Embora em problema complexos como o *MDIRPSD*, métodos exatos ainda não sejam capazes de resolver na otimalidade instâncias de médio e grande porte, nosso algoritmo *B&C* melhorou os limitantes para as instâncias clássicas do problema. Dessa maneira, trabalhos futuros que naturalmente tendem aos desenvolvimento de métodos heurísticos, metaheurísticos ou híbridos, podem ser melhor avaliados a partir dos limitantes obtidos neste estudo.

TABELA V
COMPARAÇÃO ENTRE AS POLÍTICAS *ML* E *OU* COM CUSTO DE ESTOQUE E $|T| = 3$ PERÍODOS

$\|\mathcal{T}\| = 3$			ML				OU					$(OU - ML)/ML$	
$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	$\Delta(\%)\overline{UB}$	$\Delta(\%)\overline{T(s)}$
5	2	5	1590.1	1590.1	0.0	1.0	5	1726.6	1726.6	0.0	0.0	8.6	-100.0
10	2	5	2638.4	2638.4	0.0	97.0	5	2908.6	2908.6	0.0	43.0	10.5	-55.7
15	2	4	3474.5	3457.1	0.4	1501.0	5	3899.2	3899.2	0.0	1504.0	12.3	446.7
20	3	4	4473.3	4452.5	0.5	1798.0	2	5027.1	4948.6	1.7	4420.0	12.5	145.8
25	4	2	5004.3	4761.5	5.0	5815.0	0	5641.4	5359.4	4.9	7200.0	12.6	23.8
30	4	3	5152.0	5053.1	1.8	4425.0	0	5960.7	5675.9	4.9	7200.0	15.8	62.7
35	5	1	6324.2	5984.0	4.9	6425.0	0	7762.7	6705.1	12.9	7200.0	22.6	12.1
40	5	0	6588.4	5778.0	11.5	7200.0	0	7930.8	6601.8	16.4	7200.0	22.0	0.0
45	6	0	7646.6	6683.4	11.8	7200.0	0	9595.7	7453.8	22.3	7200.0	26.5	0.0
50	6	0	8178.4	6978.3	13.5	7200.0	0	10409.7	7466.7	28.1	7200.0	28.6	0.0
Média			**5107.0**	**4737.6**	**4.9**	**4166.2**		**6086.3**	**5274.6**	**9.1**	**4916.7**	**17.2**	**8.9**

TABELA VI
COMPARAÇÃO ENTRE AS POLÍTICAS *ML* E *OU* COM CUSTO DE ESTOQUE E $|T| = 6$ PERÍODOS

$\|\mathcal{T}\| = 6$			ML				OU					$(OU - ML)/ML$	
$\|\mathcal{C}\|$	$\|\mathcal{P}\|$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	*OPT*	$\overline{UB}$	$\overline{LB}$	$\overline{GAP}(\%)$	$\overline{T(s)}$	$\Delta(\%)\overline{UB}$	$\Delta(\%)\overline{T(s)}$
5	2	3	3786.4	3718.9	1.9	3424.0	5	4284.6	4284.6	0.0	55.0	11.9	-77.8
10	2	0	5638.1	4928.3	12.5	7200.0	0	6206.6	5572.8	10.1	7200.0	10.1	0.0
15	2	1	9382.7	8658.4	7.6	6180.0	0	10303.7	9096.1	11.7	7200.0	9.8	48.5
20	3	0	10668.8	8751.3	15.5	7200.0	0	11755.4	9518.2	17.1	7200.0	10.8	0.0
25	4	0	10822.1	9532.2	11.2	7200.0	0	11874.7	10235.0	12.9	7200.0	9.8	0.0
30	4	0	13055.3	10900.6	15.1	7200.0	0	14481.1	11712.0	18.0	7200.0	11.2	0.0
35	5	0	14718.6	11650.9	20.5	7200.0	0	15788.7	12502.8	20.4	7200.0	7.1	0.0
40	5	0	16790.1	12699.3	24.3	7200.0	0	17664.4	13808.0	21.7	7200.0	5.1	0.0
45	6	0	16275.4	11515.4	29.0	7200.0	0	16839.0	13049.3	22.3	7200.0	3.6	0.0
50	6	0	18156.2	11557.2	36.2	7200.0	0	19011.9	13075.0	31.2	7200.0	4.8	0.0
Média			**11929.4**	**9391.3**	**17.4**	**6720.4**		**12821.0**	**10285.4**	**16.5**	**6485.5**	**8.4**	**-2.9**

AGRADECIMENTOS

Agradecemos ao Centro de Computação Científica e Software Livre (C3SL) da Universidade Federal do Paraná pela assistência e recursos computacionais oferecidos. Agradecemos também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro N. 1554767. Agradecemos a três revisores anônimos por suas valiosas sugestões em uma versão anterior deste artigo.

REFERÊNCIAS

[1] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, and A. Løkketangen, "Industrial aspects and literature survey: Combined inventory management and routing," vol. 37, no. 9, pp. 1515–1536, sep 2010.

[2] K. Govindan, "Vendor-managed inventory: A review based on dimensions," *International Journal of Production Research*, vol. 51, no. 13, pp. 3808–3835, jul 2013.

[3] M. D. Arango Serna, J. A. Zapata Cortes, and D. Gutierrez Sepulveda, "Modeling the inventory routing problem (irp) with multiple depots with genetic algorithms," *IEEE Latin America Transactions*, vol. 13, no. 12, pp. 3959–3965, 2015.

[4] L. C. Coelho, J.-F. Cordeau, and G. Laporte, "Thirty Years of Inventory Routing," *Transportation Science*, vol. 48, no. 1, pp. 1–19, aug 2013.

[5] L. Bertazzi, L. C. Coelho, A. De Maio, and D. Laganà, "A matheuristic algorithm for the multi-depot inventory routing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 122, pp. 524–544, feb 2019.

[6] L. C. Coelho and G. Laporte, "A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem," *International Journal of Production Research*, vol. 51, no. 23-24, pp. 7156–7169, nov 2013.

[7] ——, "The exact solution of several classes of inventory-routing problems," *Computers and Operations Research*, vol. 40, no. 2, pp. 558–565, feb 2013.

[8] M. D. Arango Serna, J. A. Zapata Cortes, and D. Gutierrez Sepulveda, "Modeling the Inventory Routing Problem (IRP) with multiple depots with genetic algorithms," *IEEE Latin America Transactions*, vol. 13, no. 12, pp. 3959–3965, 2015.

[9] T. A. Guimarães, L. C. Coelho, C. M. Schenekemberg, and C. T. Scarpin, "The two-echelon multi-depot inventory-routing problem," *Computers and Operations Research*, vol. 101, pp. 220–233, jan 2019.

[10] L. Bertazzi, G. Paletta, and M. G. Speranza, "Deterministic Order-Up-To Level Policies in an Inventory Routing Problem," *Transportation Science*, vol. 36, no. 1, pp. 119–132, jan 2003.

[11] C. Archetti, L. Bertazzi, G. Laporte, and M. G. Speranza, "A branch-and-cut algorithm for a vendor-managed inventory-routing problem," *Transportation Science*, vol. 41, no. 3, pp. 382–391, 2007.

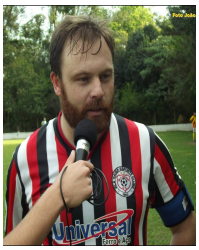
[12] L. C. Coelho and G. Laporte, "Improved solutions for inventory-routing problems through valid inequalities and input ordering," *International Journal of Production Economics*, vol. 155, pp. 391–397, 2014.

[13] J. Lysgaard, A. N. Letchford, and R. W. Eglese, "A new branch-and-cut algorithm for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 100, no. 2, pp. 423–445, Jun 2004.

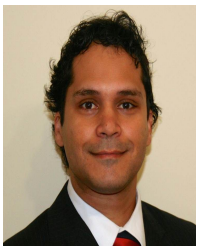
[14] M. Padberg and G. Rinaldi, "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems," *SIAM Review*, vol. 33, no. 1, pp. 60–100, mar 1991.

[15] L. C. Coelho, J. F. Cordeau, and G. Laporte, "Consistency in multi-vehicle inventory-routing," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 270–287, 2012.

Cleder M. Schenekemberg é graduado em Matemática pela Universidade do Estado de Santa Catarina (2012). Mestre (2015) e Doutor (2019) em Pesquisa Operacional pelo Programa de Pós-Graduação em Métodos Numéricos em Engenharia, área de concentração: Programação Matemática, na Universidade Federal do Paraná.



Cassius T. Scarpin é Doutor (2012) e Mestre (2007) em Pesquisa Operacional pelo programa de Pós Graduação em Métodos Numéricos em Engenharia da Universidade Federal do Paraná. é graduado em Licenciatura em Matemática (2002) e em Engenharia de Produção (2010) pela mesma universidade. Possui experiência na área de Engenharia de Produção, com interesse em Pesquisa Operacional e Logística.



José E. Pécora Jr. é Doutor em Administração de Empresas (2008), pelo departamento de operações e sistemas de decisão na Université Laval no Canadá, mestrado em Matemática Aplicada e Computacional pela Universidade Estadual de Campinas (2002) e Bacharelado em Matemática Aplicada e Computacional pela Universidade Estadual de Campinas (1998). Tem experiência na área de Pesquisa Operacional, com ênfase em Matemática Discreta e Combinatória, atuando principalmente nos seguintes temas: hibridação de métodos heurísticos com técnicas de programação inteira mista, otimização combinatória e aplicações das técnicas de pesquisa operacional à problemas industriais reais, redes logísticas, supply chain, administração da produção e gestão de serviços de saúde.

Thiago A. Guimarães é Doutor (2015) e Mestre (2011) em Pesquisa Operacional pelo programa de Pós Graduação em Métodos Numéricos em Engenharia da Universidade Federal do Paraná. é graduado em Engenharia de Produção Civil pela Universidade Tecnológica Federal do Paraná e em Ciências Econômicas pela Universidade Federal do Paraná. Atua na área de Pesquisa Operacional, com ênfase em Logística de Suprimentos e Distribuição.