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

UM ALGORITMO DE BUSCA EM VIZINHANÇA COM PROFUNDIDADE VARIÁVEL PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM COLETAS E ENTREGAS SIMULTÂNEAS

Davi Otoch Barroso
Universidade de Fortaleza, Programa de Pós-Graduação em Informática Aplicada
Av. Washigton Soares 1321, CEP: 60811-905, Fortaleza, Ceará, Brasil
davi268@edu.unifor.br

Napoleão Vieira Nepomuceno
Universidade de Fortaleza, Programa de Pós-Graduação em Informática Aplicada
Av. Washigton Soares 1321, CEP: 60811-905, Fortaleza, Ceará, Brasil
napoleaovn@unifor.br

RESUMO

Neste artigo, aborda-se o Problema de Roteamento de Veículos com Coletas e Entregas Simultâneas através de um algoritmo de Busca em Vizinhança de Profundidade Variável. Este problema tem aplicação em áreas de forte impacto econômico, social e ambiental, tais como logística reversa, reciclagem e gestão de resíduos. Neste contexto, as operações de distribuição e de recolhimento precisam ser consideradas em conjunto para minimizar os esforços logísticos. Uma nova formulação em Programação Linear Inteira (PLI) é proposta para a versão do problema com frota heterogênea. Propõe-se um método híbrido com uma estrutura de vizinhança de profundidade variável explorada através da resolução de um modelo em PLI, de maneira similar a outras hibridizações baseadas em redução da instância do problema. Os resultados dos experimentos computacionais são promissores, obtendo-se novos limites superiores (soluções viáveis) e limites inferiores para instâncias da literatura.

PALAVRAS CHAVE. Roteamento de Veículos, Busca por Vizinhança, Programação Linear Inteira.

Tópicos (Logística e Transportes, Metaheurísticas, Otimização Combinatória)

ABSTRACT

We address the Vehicle Routing Problem with Simultaneous Pickups and Deliveries by means of a Variable Depth Neighborhood Search algorithm. This problem has applications in areas with significant economic, social, and environmental impact, such as reverse logistics, recycling, and waste management. In this context, distribution and collection operations need to be considered together to minimize logistical efforts. A new Integer Linear Programming formulation is proposed for the problem version with a heterogeneous fleet. We propose a hybrid method with a variable depth neighborhood structure explored through the resolution of an ILP model, similar to other hybridizations based on problem instance reduction. The results of the computational experiments are promising, yielding new upper and lower bounds for instances from the literature.

KEYWORDS. Vehicle Routing, Neighborhood Search, Integer Linear Programming.

Paper topics (Logistics and Transportation, Metaheuristics, Combinatorial Optimization)

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

1. Introdução

Diversos trabalhos de pesquisa vêm sendo realizados para investigação de métodos de otimização na área de roteamento de veículos [Braekers et al., 2016], impulsionados particularmente pelos desafios que as companhias se deparam para aperfeiçoar seus sistemas de transporte. O Problema de Roteamento de Veículos (VRP, do inglês *Vehicle Routing Problem*) surge como o problema de projetar rotas de transporte no intuito de atender de maneira eficiente as demandas de clientes espalhados geograficamente. Considerando-se a variedade de aplicações no mundo real, o VRP pode ser visto como uma extensa classe de problemas. Um dos tipos da subclasse comumente referenciada como Problemas de Roteamento de Veículos com *Backhauls* [Parragh et al., 2008] – que trata particularmente do transporte de mercadorias de/para um depósito – é o Problema de Roteamento de Veículos com Coletas e Entregas Simultâneas (VRPSPD, do inglês *Vehicle Routing Problem with Simultaneous Pickup and Delivery*), no qual cada cliente, que demanda tanto operações de coleta e entrega, tem que ser atendido uma única vez por um mesmo veículo. O VRPSPD foi definido primeiramente em [Min, 1989] e tem recebido bastante atenção devido a sua aplicabilidade em diversos sistemas de logística reversa [Dethloff, 2001].

Na literatura científica, o VRPSPD tem tipicamente assumido uma frota homogênea de veículos [Montané e Galvão, 2002; Dell'Amico et al., 2006; Ai e Kachitvichyanukul, 2009; Goksal et al., 2013; Subramanian et al., 2013; Avci e Topaloglu, 2015; Kalayci e Kaya, 2016], e os resultados apresentados nesses artigos indicam que as instâncias investigadas estão tratáveis. Entretanto, em muitas situações práticas, as empresas empregam uma frota heterogênea para satisfazer as demandas dos clientes. De fato, contribuições científicas para o Problema de Roteamento de Veículos com Coletas e Entregas Simultâneas com frota heterogênea (HVRPSPD, do inglês *Heterogeneous Vehicle Routing Problem with Simultaneous Pickup and Delivery*) só vieram a surgir bem recentemente [Avci e Topaloglu, 2016; Nepomuceno et al., 2019; Keçeci et al., 2021; Nepomuceno et al., 2023], e a literatura científica sobre esta variante do problema ainda é relativamente escassa.

A principal contribuição deste artigo é um algoritmo de Busca em Vizinhança com Profundidade Variável (VDNS, do inglês *Variable-Depth Neighborhood Search*) [Ahuja et al., 2002; Altner et al., 2014] para tratar o VRPSPD. Além disso, estende-se uma formulação em PLI proposta em [Rieck e Zimmermann, 2013] para o VRPSPD em sua versão de frota homogênea para frota heterogênea, como uma alternativa à formulação apresentada em [Avci e Topaloglu, 2016]. A vizinhança de uma solução viável é induzida por um modelo em PLI e explorada através da aplicação de um solucionador de modelos em PLI, de maneira semelhante a hibridizações baseadas em redução da instância do problema [Blum e Raidl, 2016] — viz., *Generate & Solve* (G&S) [Saraiva et al., 2013] e *Construct, Merge, Solve & Adapt* (CMSA) [Blum et al., 2016] – ou vizinhanças baseadas em fixação de variáveis – viz., *Relaxation Induced Neighborhood Search* (RINS) [Danna et al., 2005]. Empregam-se as mesmas instâncias de benchmark de [Avci e Topaloglu, 2016] e apresentam-se novas melhores soluções e limites inferiores para estas instâncias.

O restante deste artigo está organizado da seguinte forma. A Seção 2 traz uma visão geral sobre os trabalhos relacionados a esta pesquisa. Na Seção 3, propõe-se uma nova formulação em PLI para o VRPSPD com frota heterogênea, concebe-se uma estrutura de vizinhança de profundidade variável, e introduz-se um algoritmo de VDNS para tratar o problema. Na Seção 4, resultados computacionais em instâncias de benchmark são apresentados e discutidos. A Seção 5 encerra o artigo com as considerações finais e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Esta seção traz uma visão geral do VRPSPD com frota homogênea e heterogênea. O leitor interessado é remetido a [Koç et al., 2020] para uma revisão abrangente. Além disso, faz-se uma introdução aos métodos de Busca em Vizinhança de Larga Escala (VLSN, do inglês Very Large-Scale Neighborhood Search) [Altner et al., 2014].

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

2.1. Problema de Roteamento de Veículos com Coletas e Entregas Simultâneas

Nos sistemas de logística reversa, é comum que a entrega e a coleta ocorram na mesma parada. Essa possibilidade foi introduzida na literatura por Min [1989], que investigou uma aplicação do mundo real lidando com um sistema de distribuição de bibliotecas públicas. A relação entre o VRPSPD e outros problemas de roteamento de veículos, no entanto, não foi formalmente estabelecida até muito mais tarde em [Dethloff, 2001]. Várias abordagens heurísticas e exatas têm sido propostas para lidar com o VRPSPD ao longo dos anos.

As metodologias mais recentes para o VRPSPD padrão foram publicadas em [Subramanian et al., 2013; Goksal et al., 2013; Avci e Topaloglu, 2015; Kalayci e Kaya, 2016]. Em [Subramanian et al., 2013], os autores apresentaram uma abordagem de branch-cut-and-price (BCP) que foi testada em instâncias de benchmark bem conhecidas envolvendo até 200 clientes. Quatro instâncias foram resolvidas pela primeira vez e alguns limites inferiores foram melhorados. Em [Goksal et al., 2013], os autores propuseram uma abordagem heurística baseada em otimização por enxame de partículas na qual uma busca local é realizada por um algoritmo de descida de vizinhança variável. Em Avci e Topaloglu [2015], os autores apresentaram uma abordagem de solução de busca local adaptativa que hibridiza um algoritmo inspirado em têmpera simulada com descida de vizinhança variável. Finalmente, em [Kalayci e Kaya, 2016], os autores investigaram uma metaheurística híbrida baseada em um sistema de colônia de formigas e uma busca de vizinhança variável para sua solução. Os resultados apresentados nesses artigos indicam que as instâncias de benchmark já estão bem resolvidas.

O VRPSPD heterogêneo foi abordado pela primeira vez em [Avci e Topaloglu, 2016]. Os autores apresentaram uma extensão para a versão do VRPSPD com frota homogênea baseada nas formulações de fluxo propostas em [Montané e Galvão, 2002; Ai e Kachitvichyanukul, 2009]. Em [Nepomuceno et al., 2019], os autores propuseram um algoritmo randomizado rápido que constrói soluções para o HVRPSPD, alternando probabilisticamente entre dois critérios gulosos para selecionar o próximo cliente a ser visitado: uma dada ordenação de clientes ou a estratégia do vizinho mais próximo. Para avaliar a proposta, os autores usaram as mesmas instâncias de benchmark de [Avci e Topaloglu, 2016]. A abordagem heurística melhorou várias das melhores soluções conhecidas, executando em um tempo mais curto. Em [Keçeci et al., 2021], foram propostas uma nova formulação matemática e uma abordagem heurística para abordar o problema. Os autores apresentaram resultados para as instâncias de benchmark de [Avci e Topaloglu, 2016], no entanto, seus resultados são controversos em relação aos limites inferiores obtidos em [Avci e Topaloglu, 2016; Nepomuceno et al., 2019]. Em [Nepomuceno et al., 2023], os autores propuseram diferentes estruturas de vizinhança de uma solução viável e descreveram um algoritmo de Busca em Vizinhança de Grande Escala (VLSN, do inglês *Very Large-Scale Neighborhood Search*) para abordar o problema com frota heterogênea, melhorando consistentemente os resultados para as instâncias de benchmark de [Avci e Topaloglu, 2016].

2.2. Busca em Vizinhança de Grande Escala

Em [Ahuja et al., 2002], os autores categorizam métodos de Busca em Vizinhança de Grande Escala em três classes: métodos de profundidade variável; métodos de melhoria baseados em fluxo de rede; e métodos baseados em restrição a subclasses solúveis em tempo polinomial. Em particular, a Busca em Vizinhança com Profundidade Variável explora parcialmente vizinhanças baseadas na execução simultânea de um número váriável de movimentos (por exemplo, inserção/remoção de clientes de uma rota ou troca de posições de clientes de uma rota). A classe de algoritmos VLSN é, no entanto, bastante ampla. De fato, um algoritmo de busca de vizinhança é considerado como pertencente à classe de VLSN se a vizinhança que ele busca é "muito grande" em relação ao tamanho do problema.

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

Em [Altner et al., 2014], outras classes de VLSN foram introduzidas, entre as quais destacamos as vizinhanças baseadas em fixação de variáveis. A ideia é resolver um um modelo em PLI, fixando-se um subconjunto das variáveis de decisão [Danna et al., 2005]. A proposta deste trabalho é bastante relacionada a essa ideia de explorar a vizinhança de uma solução por meio de um solucionador de modelos em PLI. Entretanto, em vez de fixar explicitamente os valores das variáveis, o modelo é definido como em outras hibridizações baseadas em redução da instância do problema.

A ideia geral da redução da instância do problema, conforme enunciado em [Blum e Raidl, 2016], é a seguinte. Se fosse possível identificar uma sub-instância substancialmente reduzida de uma dada instância do problema, de modo que a sub-instância contenha soluções de alta qualidade para a instância original do problema, poder-se-ia aplicar uma técnica exata com um esforço computacional razoável para obter uma solução de alta qualidade para a instância original. Dada uma instância de problema $\mathcal{I}$ para um problema $\mathcal{P}$, defini-se o conjunto C como o conjunto de todos os componentes possíveis dos quais as soluções para a instância do problema são compostas. Uma solução viável s para $\mathcal{I}$ induz um subconjunto dos componentes $C_s \subseteq C$. Além disso, um conjunto $C' \subseteq C$ define indiretamente uma sub-instância reduzida de $\mathcal{I}$.

Propõe-se, neste trabalho, que C' seja escolhido de maneira sistemática para definir uma vizinhança $\eta(s)$ de uma solução viável s. Como exemplo, no caso do VRPSPD, o conjunto C é formado por todos os arcos que interligam os nós da instância. Os arcos pertencentes a uma solução viável s – ou seja, o conjunto de rotas de veículos – formam o conjunto de componentes C_s. À medida que adicionamos outros arcos além dos apresentados em s para obter C', definimos uma sub-instância reduzida do VRPSPD cujo espaço de busca de solução é limitado a soluções que usam apenas arcos apresentados em C'. Uma vizinhança $\eta(s)$ poderia ser definida, por exemplo, para adicionar os arcos que formam uma clique entre um dado subconjunto de nós.

3. Proposta

Nesta seção, apresenta-se uma extensão da formulação matemática proposta em [Rieck e Zimmermann, 2013]. Em seguida, concebe-se uma estrutura de vizinhança de profundidade variável, explorada por meio da resolução de um modelo em PLI, e propõe-se um algoritmo de Busca em Vizinhança com Profundidade Variável para tratar o problema.

3.1. Formulação Matemática

O HVRPSPD pode ser formalmente enunciado como: Seja $G = (N_0 = N \cup \{0\}, A)$ um grafo completo direcionado, onde o conjunto de nós N representa os clientes e o nó 0 representa o depósito. Cada cliente $i \in N$ possui uma demanda de entrega d_i e uma demanda de coleta p_i não negativas. Cada arco $(i, j) \in A$ está associado a uma distância c_{ij} do nó i ao nó j. Cada tipo de veículo $k \in K$ está associado a um custo fixo f^k, um custo variável por unidade de distância g^k e uma capacidade q^k, e não existe nenhuma restrição quanto ao número de veículos de cada tipo. O objetivo é determinar a composição da frota e o conjunto de rotas que minimiza o custo total, de tal forma que: (1) cada veículo perfaz apenas uma rota; (2) cada rota começa e termina no depósito; (3) cada cliente é visitado apenas uma vez por um dos veículos; (4) cada veículo satisfaz completamente as demandas de coleta e entrega dos clientes em sua rota; e (5) a carga de um veículo não excede sua capacidade em nenhum momento.

Definem-se as variáveis de decisão $x_{ij}^k \in \{0, 1\}$, indicando se o arco (i, j) é atravessado por um veículo do tipo k (valor 1), ou não (valor 0). Definem-se ainda as variáveis de decisão $y_{ij} \geq 0$, representando a quantidade de carga ainda a ser entregue aos clientes da rota no momento em que o veículo está atravessando o arco (i, j), e as variáveis de decisão $z_{ij} \geq 0$, representando a quantidade de carga já coletada dos clientes da rota no momento em que o veículo está atravessando o arco (i, j). O problema de otimização pode então ser formulado:

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

$$\min \sum_{j \in N} \sum_{k \in K} f^k x_{0j}^k + \sum_{i \in N_0} \sum_{j \in N_0 : i \neq j} \sum_{k \in K} c_{ij} g^k x_{ij}^k \tag{1}$$

$$\text{s.t.} \sum_{i \in N_0 : i \neq v} \sum_{k \in K} x_{iv}^k = 1 \qquad \forall v \in N \tag{2}$$

$$\sum_{i \in N_0 : i \neq v} x_{iv}^k - \sum_{j \in N_0 : j \neq v} x_{vj}^k = 0 \qquad \forall v \in N, \forall k \in K \tag{3}$$

$$\sum_{i \in N_0 : i \neq v} y_{iv} - \sum_{j \in N_0 : j \neq v} y_{vj} = d_v \qquad \forall v \in N \tag{4}$$

$$\sum_{j \in N_0 : j \neq v} z_{vj} - \sum_{i \in N_0 : i \neq v} z_{iv} = p_v \qquad \forall v \in N \tag{5}$$

$$y_{ij} + z_{ij} \leq \sum_{k \in K} (q^k - max(0, d_i - p_i, p_j - d_j)) x_{ij}^k \qquad \forall i \in N_0, \forall j \in N_0 : i \neq j \tag{6}$$

$$y_{ij} \geq d_j \sum_{k \in K} x_{ij}^k \qquad \forall i \in N_0, \forall j \in N_0 : i \neq j \tag{7}$$

$$z_{ij} \geq p_i \sum_{k \in K} x_{ij}^k \qquad \forall i \in N_0, \forall j \in N_0 : i \neq j \tag{8}$$

$$y_{ij} \leq \sum_{k \in K} (q^k - d_i) x_{ij}^k \qquad \forall i \in N_0, \forall j \in N_0 : i \neq j \tag{9}$$

$$z_{ij} \leq \sum_{k \in K} (q^k - p_j) x_{ij}^k \qquad \forall i \in N_0, \forall j \in N_0 : i \neq j \tag{10}$$

$$\sum_{i \in N_0 : i \neq v} y_{iv} \geq d_v \qquad \forall v \in N \tag{11}$$

$$\sum_{j \in N_0 : j \neq v} z_{vj} \geq p_v \qquad \forall v \in N \tag{12}$$

$$x_{ij}^k \in \{0, 1\}, y_{ij}, z_{ij} \in \mathbb{R} \qquad \forall i \in N_0, \forall j \in N_0, \forall k \in K \tag{13}$$

A função objetivo (1) consiste em minimizar o custo fixo e variável de utilização dos veículos. As restrições (2) garantem que cada cliente é visitado uma única vez. As restrições (3) caracterizam a conservação de fluxo. As restrições (4) e (5) são as equações dos saldos relativos ao atendimento das demandas de entrega e coleta do cliente v, e garantem a inexistência de subciclos. As restrições (6) garantem que a capacidade de um veículo não seja excedida em nenhuma rota. As restrições (7) impõem que a quantidade ainda a ser entregue ao atravessar o arco (i, j) deve ser pelo menos igual à demanda de entrega do cliente j. De forma análoga, as restrições (8) impõem que a quantidade já coletada ao atravessar o arco (i, j) deve ser pelo menos igual à demanda de coleta do cliente i. As restrições (9) garantem que a quantidade ainda a ser entregue aos clientes da rota ao passar pelo arco (i, j) seja menor ou igual à capacidade do veículo menos a demanda de entrega do cliente i. De maneira análoga, as restrições (10) garantem que a quantidade já coletada ao passar pelo arco (i, j) seja menor ou igual à capacidade do veículo menos a demanda de coleta do cliente j. Ressalta-se que as restrições (7)-(10) são utilizadas para restringir mais o domínio das variáveis contínuas do modelo, conforme Subramanian et al. [2010]. As restrições (11) e (12), que estabelecem limites mínimos para as quantidades de carga ao chegar e ao sair de um cliente v, são utilizadas para deixar a formulação mais forte [Rieck e Zimmermann, 2013]. Por fim, as restrições (13) determinam o domínio das variáveis de decisão.

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

Esta formulação, em comparação àquela proposta em [Avci e Topaloglu, 2016], evita que as variáveis de decisão x possuam um índice para cada veículo individualmente, mas sim um índice sobre cada tipo de veículo. A formulação anterior exigia um limite arbitrário, geralmente grande, do tamanho da frota de veículos. Ainda, as variáveis de decisão y e z não estão associadas a um veículo, nem sequer a um tipo de veículo, diminuindo ainda a dimensão do modelo em PLI.

3.2. Estrutura de Vizinhança

Considere uma solução viável s para o HVRPSPD. Na Figura 1, esta solução é representada como o conjunto de rotas começando e terminando no depósito (nó 0) e visitando conjuntos disjuntos de clientes. Se apenas os arcos apresentados na solução s são incluídos em uma subinstância reduzida, a otimização fica então restrita à escolha do tipo de veículo para cada rota. Porém, à medida que são incluídos outros arcos além daqueles apresentados na solução s, o número de soluções viáveis pode crescer exponencialmente. Note-se que, se o grafo completo é considerado, resolve-se a instância original do problema de forma exata.

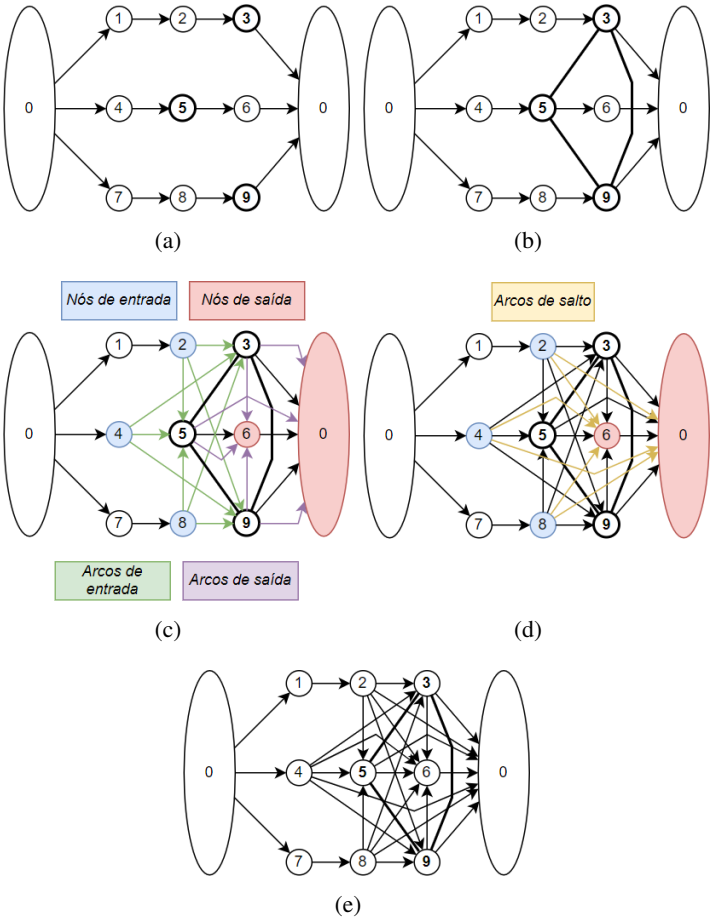


Figura 1: Estrutura de vizinhança: (a) escolha de clientes da clique; (b) inclusão dos arcos da clique; (c) inclusão dos arcos de entrada e saída da clique; (d) inclusão dos arcos de salto da clique; e (e) arcos da estrutura de vizinhança.

Na definição da estrutura de vizinhança proposta, conforme ilustrado na Figura 1, partindo-se de uma solução viável inicial, escolhe-se um subconjunto de nós cujo tamanho está associado à profundidade da vizinhança do algoritmo VDNS. Adicionam-se, além dos arcos da solução atual, os

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

arcos para formar uma clique dos nós selecionados. Definem-se os nós de entrada (saída) da clique como aqueles que precedem (sucedem) imediatamente um nó da clique em sua rota. Adicionam-se, então, os arcos entre os nós de entrada e os nós da clique, e os arcos entre os nós da clique e os nós de saída. Finalmente, adicionam-se os arcos entre os nós de entrada e os nós de saída da clique.

3.3. Algoritmo de Busca em Vizinhança com Profundidade Variável

Em cada iteração do Algoritmo 1, considerando a solução atual s, uma nova sub-instância reduzida é tratada. Na prática, uma instanciação μ do modelo em PLI (1–13) que considera, no entanto, apenas os arcos usados para montar as rotas da solução atual s e os arcos definidos pela estrutura de vizinhança η é gerada e explorada por meio de um solucionador de modelos em PLI. Para melhorar o desempenho do solucionador, usamos s para fornecer uma solução inicial (*warm start*) para μ, que pode ser usada para podar nós da árvore de busca. A eficácia desta inicialização depende em grande parte de quão boa é a solução s fornecida e, portanto, pode ser muito útil à medida que esta se aproxima da solução ótima global. Esta estratégia também garante que o solucionador sempre retorne uma solução que seja pelo menos tão boa quanto a solução atual. O algoritmo continua a iterar até que um limite de tempo de computação τ_{vdns} seja atingido ou um número máximo de iterações consecutivas δ sem melhoria seja alcançado.

Algoritmo 1: Pseudocódigo do algoritmo VDNS.

```
construir uma solução viável s
while critério de parada não satisfeito do
    selecionar um subconjunto de nós para formar a clique de tamanho κ
    construir os arcos definidos pela estrutura de vizinhança η
    gerar o modelo μ associado à sub-instância reduzida induzida por s e η
    resolver o modelo μ assumindo-se um limite de tempo τ_solver
    atualizar s para ser a solução encontrada para o modelo μ
    deixar γ ser o gap de otimalidade na execução do modelo μ
    if s é uma nova solução then
        aplicar operadores de busca local em s
    end
    atualizar o tamanho da clique κ de acordo com γ
end
return s
```

A profundidade da estrutura de vizinhança, ou tamanho κ do subconjunto de nós para formar a clique, varia de acordo com o *gap* de otimalidade γ obtido pelo solucionador de modelos em PLI dentro do limite de tempo τ_{solver} estabelecido para explorar a vizinhança atual. Como grande parte do tempo do solucionador é dedicada a provar a otimalidade da solução corrente, isso garante um melhor equilíbrio entre consumo de tempo e qualidade da solução da sub-instância. O ponto é que, quanto mais próximo de 0 o valor de γ for, mais bem-sucedido o solucionador terá se mostrado ao lidar com a sub-instância reduzida atual do modelo em PLI. Portanto, decidiu-se aumentar o valor de κ em 1 unidade sempre que o solucionador retornar um *gap* de otimalidade menor ou igual a 1%, e diminuir o valor de κ em 1 unidade sempre que o solucionador retornar um *gap* de otimalidade maior que 1%. Sempre que uma nova solução é encontrada, aplicam-se ainda três operadores de busca local comumente empregados na resolução de VRPs (*2-Opt*, *Customer Insertion*, *Customer Interchange*) [Gajpal e Abad, 2009].

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

4. Experimentos Computacionais

Para os experimentos computacionais, foi utilizada uma máquina PC Intel Core i7-10700 2.9GHz com 16GB. A linguagem de programaçaão utilizada foi Java com a biblioteca IBM ILOG CPLEX 22.1.0. Para obter uma solução inicial, fez-se uso de uma implementação do algoritmo proposto em [Nepomuceno et al., 2023], juntamente com os operadores de busca local (*2-Opt*, *Customer Insertion*, *Customer Interchange*) [Gajpal e Abad, 2009].

Instâncias de benchmark disponíveis em [Avci e Topaloglu, 2016] foram usadas em nosso estudo computacional. Essas instâncias estão agrupadas em dois conjuntos, cada um deles consistindo em 14 instâncias. O primeiro conjunto contém instâncias de problemas com 10-100 clientes. Os valores das coordenadas dos clientes estão distribuídos uniformemente no intervalo $[0, 50]$, enquanto as demandas de coleta e entrega dos clientes estão distribuídas uniformemente no intervalo $[0, 100]$. O número de tipos de veículos varia de 2 a 4, a capacidade dos veículos de 150 a 900, o custo fixo de um veículo de 40 a 150, e o custo variável por unidade de distância de 1.00 a 1.20. O segundo conjunto consiste em instâncias de problemas de maior porte com 150-550 clientes. Os valores das coordenadas dos clientes estão distribuídos uniformemente no intervalo $[0, 100]$, enquanto as demandas de coleta e entrega dos clientes estão distribuídas uniformemente no intervalo $[0, 50]$. O número de tipos de veículos varia de 2 a 3, a capacidade dos veículos de 500 a 2100, o custo fixo de um veículo de 75 a 290, e o custo variável por unidade de distância de 1.00 a 1.12.

Para calibração do algoritmo VDNS, concentraram-se os esforços na determinação do valor de τ_{solver} porque o algoritmo demonstrou-se ser especialmente sensível a ele. Note que, quanto maior o valor de τ_{solver}, melhor explora-se uma vizinhança de uma solução, mas menor é o número de iterações do algoritmo VDNS. Além disso, os outros parâmetros do algoritmo que não estão relacionados aos critérios de parada dependem amplamente do valor de τ_{solver} e são adaptados de acordo com ele dinamicamente. Após os testes de calibração, definiu-se o valor limite de tempo do solver $\tau_{solver} = 20$ s, assumindo-se um limite de tempo do algoritmo $\tau_{vlsn} = 3600$ s e um número $\delta = 200$ de iterações consecutivas sem melhoria como critérios de parada. Assumiu-se ainda um valor inicial de $\kappa = 10$. Note que, se a instância tiver no máximo 10 nós, resolve-se diretamente o modelo em PLI da instância.

4.1. Resultados Comparativos

Os resultados numéricos para as instâncias de benchmark propostas em [Avci e Topaloglu, 2016] são apresentados na Tabela 1, juntamente à caracterização das instâncias (onde n e t representam respectivamente o número de clientes e o número de tipos de veículos). Para cada algoritmo, exibimos o melhor valor de solução, o valor médio de solução e o tempo médio de computação (ao longo de 10 execuções). As melhores soluções conhecidas são destacadas em negrito.

O software IBM ILOG CPLEX 22.1.0 foi empregado como resolvedor dos modelos em PLI com todas as configurações mantidas em seus padrões, exceto o limite de tempo de execução. Os resultados obtidos são comparados com aqueles registrados pelo algoritmo HLS [Avci e Topaloglu, 2016] e pelo algoritmo VLSN [Nepomuceno et al., 2023]. Além disso, apresentam-se os resultados obtidos pelo aplicação direta do CPLEX para a nova formulação matemática (1–13) proposta para o problema, onde *lb*, *ub*, *gap* e *time* representam, respectivamente, o limite inferior, o limite superior, o gap de otimalidade e o tempo de execução do resolvedor.

Os resultados numéricos demonstram que o algoritmo VDNS supera os métodos anteriores, obtendo melhores soluções para praticamente todas as instâncias, exceto a instância *1-08*. Além disso, o algoritmo VDNS foi capaz de encontrar 7 novas melhores soluções para o conjunto de instâncias de tamanho menor. Para o conjunto de instâncias de tamanho maior, o algoritmo VDNS foi capar de encontrar novas melhores soluções para todas as instâncias. Destaca-se ainda que, com a nova formulação matemática (1–13), foram encontrados novos limites inferiores para 23 (de 28) instâncias, e comprovada a otimalidade da solução da instância *1-04*.

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

Tabela 1: Resultados comparativos para instâncias de *benchmark* propostas em [Avci e Topaloglu, 2016].

			CPLEX				HLS[1]			VLSN[2]			VDNS[3]		
inst.	n	t	lb	ub	gap(%)	time(s)	best	avg.	time(s)	best	avg.	time(s)	best	avg.	time(s)
1-01	10	2	620,2	**620,2**†	0,0	0,8	**620,2**†	620,2	17,2	**620,2**†	620,2	471,8	**620,2**†	620,2	256,4
1-02	10	2	588,5	**588,5**†	0,0	9,1	**588,5**†	588,5	14,7	**588,5**†	588,5	177,4	**588,5**†	588,5	770,4
1-03	15	3	445,1	**445,1**†	0,0	210,3	**445,1**†	445,1	22,7	**445,1**†	445,1	815,2	**445,1**†	445,1	2922,8
1-04	15	4	437,1$^{\uparrow}$	**437,1**†	0,0	474,4	**437,1**†	437,1	24,5	**437,1**†	438,5	1303,5	**437,1**†	437,1	2931,9
1-05	20	3	479,2$^{\uparrow}$	**494,0**	3,0	3600,0	**494,0**	498,9	27,1	**494,0**	494,3	1948,8	**494,0**	494,0	2890,1
1-06	20	4	526,8$^{\uparrow}$	575,5	8,5	3600,0	**542,7**	551,9	26,7	575,5	575,5	1059,0	**542,7**	542,7	2945,3
1-07	35	3	1019,8$^{\uparrow}$	1137,9	10,4	3600,0	1108,2	1123,4	56,6	1097,3	1135,7	2587,0	**1092,2***	1095,1	3087,0
1-08	35	3	1474,4$^{\uparrow}$	1661,3	11,2	3600,0	**1586,5**	1601,2	54,7	1598,5	1629,0	2309,0	1598,5	1614,2	3052,9
1-09	50	3	858,4$^{\uparrow}$	1026,9	16,4	3600,0	964,4	990,2	91,4	937,0	955,4	2927,4	**932,4***	944,8	3162,1
1-10	50	2	1036,8	1376,9	24,7	3600,0	1197,7	1228,6	95,8	1169,2	1202,8	3456,0	**1155,2***	1172,1	3501,9
1-11	75	3	1373,7$^{\uparrow}$	1780,1	22,8	3600,0	1642,2	1673,9	143,8	1531,9	1558,9	3229,7	**1493,1***	1519,6	3389,3
1-12	75	2	861,3$^{\uparrow}$	1055,5	18,4	3600,0	973,1	1002,5	164,9	925,6	960,6	3076,4	**911,5***	929,5	3207,6
1-13	100	2	1122,3$^{\uparrow}$	1430,6	21,6	3600,0	1299,5	1353,5	288,5	1314,5	1331,7	3579,9	**1226,0***	1267,6	3600,0
1-14	100	2	1322,2$^{\uparrow}$	1953,9	32,3	3600,0	1658,2	1678,2	310,3	1539,4	1563,1	3600,0	**1510,5***	1552,5	3600,0
2-01	150	3	1243,7$^{\uparrow}$	5696,7	78,2	3600,0	1499,4	1624,5	592,5	1529,6	1579,6	3600,0	**1454,9***	1478,8	3514,3
2-02	150	3	1814,5$^{\uparrow}$	4040,0	55,1	3600,0	2144,8	2152,5	548,9	2313,4	2364,0	3600,0	**2111,7***	2182,4	3600,0
2-03	200	3	2748,4$^{\uparrow}$	14567,5	81,1	3600,0	3673,1	3688,8	831,6	3576,4	3647,2	3600,0	**3289,1***	3313,5	3600,0
2-04	200	2	1838,8$^{\uparrow}$	4523,9	59,4	3600,0	2485,3	2682,5	919,4	2367,8	2435,3	3600,0	**2184,3***	2204,9	3600,0
2-05	250	3	2025,2$^{\uparrow}$	61332,8	96,7	3600,0	2639,6	2810,0	1448,1	2597,4	2651,8	3600,0	**2402,5***	2434,3	3600,0
2-06	250	2	1872,7$^{\uparrow}$	40529,8	95,4	3600,0	2549,8	2605,6	1521,3	2397,1	2500,0	3600,0	**2192,3***	2242,2	3600,0
2-07	300	3	2048,7$^{\uparrow}$	16745,4	87,8	3600,0	3205,0	3431,2	2440,4	2819,8	2880,6	3600,0	**2519,6***	2576,8	3600,0
2-08	300	2	2049,0$^{\uparrow}$	15893,8	87,1	3600,0	3252,8	3364,6	2518,2	2807,7	2862,2	3600,0	**2543,0***	2602,2	3484,3
2-09	350	3	2708,3$^{\uparrow}$	30891,1	91,2	3600,0	3457,9	3637,0	3770,1	3874,1	3966,2	3600,0	**3325,6***	3421,9	3600,0
2-10	350	2	1884,3$^{\uparrow}$	20095,3	90,6	3600,0	3760,9	3897,5	3988,7	2578,0	2617,2	3600,0	**2345,4***	2410,1	3260,2
2-11	400	2	2768,7$^{\uparrow}$	122160,7	97,7	3600,0	5809,5	6018,9	5662,4	3703,7	3808,0	3600,0	**3331,2***	3386,2	3600,0
2-12	400	2	2488,3$^{\uparrow}$	120045,6	97,9	3600,0	4045,9	4447,0	5791,5	3355,2	3425,7	3600,0	**2935,5***	3017,4	3552,3
2-13	500	2	6401,2$^{\uparrow}$	194935,8	96,7	3600,0	11008,8	12062,5	7200,0	8546,4	8627,4	3600,0	**7606,4***	7745,8	3600,0
2-14	550	2	-	-	-	3600,0	12762,0	13046,3	7200,0	10120,9	10373,9	3600,0	**9226,4***	9326,6	3600,0

[1] HLS: *adaptive hybrid local search* by Avci e Topaloglu [2016]; Testes realizados em um PC Pentium 4 3,00 GHz.
[2] MILP-VLSN: *very large-scale neighborhood search* by [Nepomuceno et al., 2023]; Testes realizados em um Intel Core i7-7500 2,7 GHz.
[3] VDNS: *variable-depth neighborhood search*; Testes realizados em um Intel Core i7-10700 2.9GHz.
* Nova melhor solução encontrada para a instância.
$^{\uparrow}$ Novo limite inferior encontrado para a instância.
† Solução ótima da instância.

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

Ressalta-se também que, ainda que fosse dada ao resolvedor uma solução inicial, a aplicação direta do CPLEX não superaria os resultados obtidos pelo algoritmo VDNS proposto, conforme ilustrado na Tabela 2.

Tabela 2: Resultados comparativos com *warm start* do CPLEX.

inst.	start	CPLEX	VDNS	inst.	start	CPLEX	VDNS
1-01	620,2	**620,2**	**620,2**	2-01	1516,9	1505,0	**1454,9**
1-02	588,5	**588,5**	**588,5**	2-02	2250,7	2250,7	**2111,7**
1-03	445,1	**445,1**	**445,1**	2-03	3360,7	3360,7	**3289,1**
1-04	446,5	**437,1**	**437,1**	2-04	2242,0	2242,0	**2184,3**
1-05	499,4	496,91	**494,0**	2-05	2500,2	2500,0	**2402,5**
1-06	565,9	**542,7**	**542,7**	2-06	2308,8	2307,4	**2192,3**
1-07	1103,3	1103,3	**1092,2**	2-07	2658,1	2656,1	**2519,6**
1-08	1636,1	1613,9	**1598,5**	2-08	2664,4	2664,4	**2543,0**
1-09	954,9	954,9	**932,4**	2-09	3550,2	3550,2	**3325,6**
1-10	1220,6	1199,1	**1155,2**	2-10	2431,6	2431,6	**2345,4**
1-11	1534,3	1530,1	**1493,1**	2-11	3491,2	3489,3	**3331,2**
1-12	935,6	935,6	**911,5**	2-12	3082,6	3082,6	**2935,5**
1-13	1341,8	1323,7	**1226,0**	2-13	7777,6	7777,6	**7606,4**
1-14	1610,0	1609,1	**1510,5**	2-14	9481,6	9481,6	**9226,4**

5. Conclusão

Neste artigo, um algoritmo de Busca em Vizinhança com Profundidade Variável foi proposto para lidar com o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas. Uma estrutura de vizinhança de uma dada solução explorada por meio de um resolvedor de modelos em PLI, é empregada dentro de um método de busca local. Além disso, um nova formulação matemática foi proposta para a variante com frota heterogênea.

Utilizando instâncias de benchmark da literatura, demonstrou-se o potencial desta proposta através de estudos comparativos com métodos atuais da literatura sobre o problem investigado. Os resultados revelam que o algoritmo VDNS proposto consistentemente superou os algoritmos concorrentes. Obtivemos novas melhores soluções para 21 (de 28) instâncias do problema, e novos limites inferiores foram obtidos para 23 (de 28) instâncias.

Como trabalho futuro, deseja-se aplicar a proposta para abordar outras variantes do VRP. De fato, tendo em mãos a formulação matemática do problema de outra variante do VRP, a aplicação da estrutura de vizinhança proposta seria direta e, portanto, uma direção natural de pesquisa.

Referências

Ahuja, R. K., Ergun, Ö., Orlin, J. B., e Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1):75–102.

Ai, T. J. e Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5):1693–1702.

Altner, D. S., Ahuja, R. K., Ergun, Ö., e Orlin, J. B. (2014). *Very Large-Scale Neighborhood Search*, p. 339–367. Springer US.

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

Avci, M. e Topaloglu, S. (2015). An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. *Computers & Industrial Engineering*, 83: 15 – 29.

Avci, M. e Topaloglu, S. (2016). A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications*, 53:160–17.

Blum, C., Pinacho, P., López-Ibáñez, M., e Lozano, J. A. (2016). Construct, merge, solve & adapt: A new general algorithm for combinatorial optimization. *Computers & Operations Research*, 68: 75–88.

Blum, C. e Raidl, G. R. (2016). *Hybridization Based on Problem Instance Reduction*, p. 45–62. Springer International Publishing.

Braekers, K., Ramaekers, K., e Nieuwenhuyse, I. V. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Danna, E., Rothberg, E., e Pape, C. L. (2005). Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90. ISSN 1436-4646.

Dell'Amico, M., Righini, G., e Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2): 235–247.

Dethloff, J. (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, 23(1):79–96.

Gajpal, Y. e Abad, P. (2009). An ant colony system (acs) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36(12):3215–3223.

Goksal, F. P., Karaoglan, I., e Altiparmak, F. (2013). A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, 65(1):39–53.

Kalayci, C. B. e Kaya, C. (2016). An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications*, 66:163–175.

Keçeci, B., Altıparmak, F., e Kara, İ. (2021). A mathematical formulation and heuristic approach for the heterogeneous fixed fleet vehicle routing problem with simultaneous pickup and delivery. *Journal of Industrial & Management Optimization*, 17(3):1069–1100.

Koç, Ç., Laporte, G., e Tükenmez, İ. (2020). A review of vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, 122:1–15.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377–386.

Montané, F. T. e Galvão, R. D. (2002). Vehicle routing problems with simultaneous pick-up and delivery service. *OPSEARCH*, 39(1):19–33.

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br

Nepomuceno, N., Saboia, R. B., e Coelho, A. L. V. (2023). A MILP-based very large-scale neighborhood search for the heterogeneous vehicle routing problem with simultaneous pickup and delivery. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, p. 330–338, New York, NY, USA. Association for Computing Machinery.

Nepomuceno, N., Saboia, R. B., e Pinheiro, P. R. (2019). A fast randomized algorithm for the heterogeneous vehicle routing problem with simultaneous pickup and delivery. *Algorithms*, 12 (8):158.

Parragh, S. N., Doerner, K. F., e Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51.

Rieck, J. e Zimmermann, J. (2013). Exact solutions to the symmetric and asymmetric vehicle routing problem with simultaneous delivery and pick-up. *Business Research*, 6(1):77–92.

Saraiva, R. D., Nepomuceno, N. V., e Pinheiro, P. R. (2013). The generate-and-solve framework revisited: Generating by simulated annealing. In Middendorf, M. e Blum, C., editors, *Evolutionary Computation in Combinatorial Optimization*, p. 262–273. Springer Berlin Heidelberg.

Subramanian, A., Uchoa, E., e Ochi, L. S. (2010). New lower bounds for the vehicle routing problem with simultaneous pickup and delivery. In Festa, P., editor, *Experimental Algorithms*, p. 276–287. Springer Berlin Heidelberg.

Subramanian, A., Uchoa, E., Pessoa, A. A., e Ochi, L. S. (2013). Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, 7(7):1569–1581.

DOI: 10.59254/sbpo-2024-193358
https://proceedings.science/p/193358?lang=pt-br