Roteamento com restrições temporais: formulação IP, estratégias algorítmicas e casos polinomiais em grafos

Thailsson Clementino[1], **Rosiane de Freitas**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas

{thailsson.clementino, rosiane}@icomp.ufam.edu.br

Abstract. *This research consisted of investigating the Vehicle Routing Problem (VRP) with time constraints, seeking to incorporate the dynamic characteristics of the express delivery logistics chain. Theoretical aspects of modeling and algorithmic strategies were explored. A formulation in Integer Linear Programming (ILP) was proposed for the VRPRDD, an adaptation of instances of Brazilian cities for the VRPTW and an analysis of polynomial cases of VRPs involving special classes of graphs, such as paths, stars, subdivided stars and trees. Dynamic programming algorithms and binary search were applied, and their runtime complexities were determined.*

Resumo. *Esta pesquisa consistiu na investigação do Problema de Roteamento de Veículos (VRP) com restrições temporais, buscando incorporar as características dinâmicas da cadeia logística de entregas expressas. Foram explorados aspectos teóricos de modelagem e estratégias algorítmicas. Foi proposta uma formulação em Programação Linear Inteira (PLI) para o VRPRDD, uma adaptação de instâncias de cidades brasileiras para o VRPTW e realizada uma análise de casos polinomiais de VRPs que envolvem classes especiais de grafos, tais como caminhos, estrelas, estrelas subdivididas e árvores. Algoritmos de programação dinâmica e busca binária foram aplicados, sendo determinadas suas complexidades de tempo de execução.*

1. Introdução

Os problemas de roteamento de veículos (VRPs) com restrições temporais associadas são um desafio complexo enfrentado por muitas empresas em todo o mundo. Ele surge quando é necessário planejar rotas para uma frota de veículos que precisam atender a um conjunto de clientes em horários específicos, levando em consideração diversas restrições, como limites de tempo de serviço dos motoristas, janelas de tempo para entrega e coleta de mercadorias, entre outras.

Esse problema é especialmente crítico para empresas de logística e transporte, que precisam entregar produtos aos clientes de forma rápida e eficiente, evitando atrasos ou erros na programação das entregas. Além disso, a otimização das rotas pode levar a uma redução significativa nos custos operacionais. Resolver esse tipo de problema é desafiador devido a dificuldades teóricas e de complexidade computacional. Estratégias como algoritmos heurísticos, meta-heurísticos, de otimização multiobjetivo e programação matemática têm sido propostas para lidar com esses desafios. A escolha da melhor estratégia depende das características específicas do problema em questão.

Esta pesquisa de iniciação científica foca no estudo de VRPs com restrições temporais, explorando a modelagem teórica e a resolução algorítmica. A seguir as **contribuições** elaboradas durante o estudo:

- Foi elaborado um compêndio para revisar VRPs com restrições temporais, incluindo classificação, formulações IP, estratégias algorítmicas e o uso de *Solvers*.
- Foi proposta uma formulação em Programação Inteira (PI) para o VRPRDD que incorpora restrições temporais de *scheduling*.
- Foram criadas estratégias exatas e heurísticas, incluindo métodos *Branch-and-Cut* (BC) e *Branch-Cut-and-Price* (BCP), solvers CPLEX e VRPSolver. Propôs-se uma formulação usando VRPSolver para resolver VRPRDD. **Apresentado no CLAIO 2022, e a ser publicado**.
- Adaptaram-se e resolveram-se instâncias para o VRPTW de entregas urbanas em cidades brasileiras de diferentes regiões, usando algoritmo *Branch-Cut-and-Price* com VRPSolver para até 350 clientes. Resultados **publicados em dois eventos**, um nacional (**III WBCI/CSBC** [Clementino et al. 2022a]) e outro latino-americano (**SLIOIA/CLEI 2022** [Clementino et al. 2022b]).
- Foi realizado um estudo, refinamento e proposição de algoritmos, com análise de complexidade, para casos polinomiais de VRPs envolvendo classes especiais de grafos (caminho, estrela, estrela subdividida). **A ser publicado**.

2. Formulação em Programação Inteira para o VRPRDD: restrições temporais de scheduling

Apesar de mais de 10 trabalhos abordarem VRPs com restrições de *scheduling*, a maioria usa metaheurísticas. Apenas dois apresentam soluções exatas: [Sun et al. 2022] e [Yang et al. 2021]. Nesta seção, serão apresentadas soluções em PLI e usando o VRPSolver para resolver o VRPRDD, ambas criadas durante um curso online na Universidade Federal Fluminense (UFF) sobre algoritmos avançados de geração de colunas e cortes.

2.1. Problema de Roteamento de Veículos com *Release Dates* e *Deadline* - VRPRDD

Dado um grafo direcionado completo $G = (V, A)$, onde o conjunto dos vértices V é composto por um vértice 0, que representa o depósito, e um conjunto de vértices $N = \{1, ..., |N|\}$, que representa os clientes que irão ser visitados. O conjunto $A = \{(i, j) \in V \times V : i \neq j\}$ são as arestas que ligam os clientes. Associado a cada aresta está um custo de viagem c_{ij} e um tempo de viagem $t_{ij} > 0$, onde t_{ij} inclui o tempo de serviço no vértice i. Em cada vértice $i \in V$ é associado uma demanda q_i, uma *release date* r_i e um *deadline* d_i.

O conjunto de veículos disponíveis para realizarem as entregas é denotado por $K = \{1, .., |K|\}$. Os veículos são homogêneos e possuem uma capacidade Q. Uma rota viável contendo o conjunto $S \subseteq N$ de clientes é definida quando $\sum_{i \in S} q_i \leq Q$ e nenhum dos clientes é visitado mais de uma vez, além de respeitar as restrições de *release date* e *deadline*. Uma solução para o VRPRDD consiste em $|K|$ rotas viáveis, uma para cada veículo $k \in K$. As rotas $R_1, R_2, ..., R_{|K|}$ e os correspondentes *clusters* $S_1, S_2, ..., S_{|K|}$ fornecem uma solução viável para o VRPRDD se todas as rotas forem viáveis e os *clusters* formarem uma partição de N. Nosso objetivo é fornecer uma solução viável com o menor somatório possível dos custos das rotas.

2.2. Formulação Linear Inteira

Com base em pesquisas anteriores sobre VRPTW [Desaulniers et al. 2014], realizou-se a modelagem do VRPRDD por meio de PLI. Para isso, introduziu-se a variável binária de três índices x_{ijk} usada para indicar se a aresta (i, j) foi utilizada pelo veículo k. Além disso a variável inteira T_{ik} especifica o tempo em que o cliente i é servido pelo cliente k. A formulação é apresentada a seguir:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{1a}$$

$$\text{Sujeito a} \quad \sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ijk} = 1 \qquad \forall i \in N \tag{1b}$$

$$\sum_{j \in \delta^+(0)} x_{0jk} = 1 \qquad \forall k \in K \tag{1c}$$

$$\sum_{i \in \delta^-(j)} x_{ijk} - \sum_{i \in \delta^+(j)} x_{jik} = 0 \qquad \forall k \in K, j \in N \tag{1d}$$

$$\sum_{i \in \delta^-(n+1)} x_{i,n+1,k} = 1 \qquad \forall k \in K \tag{1e}$$

$$\sum_{i \in N} q_i \sum_{j \in \delta^+(i)} x_{ijk} \leq Q \qquad \forall k \in K \tag{1f}$$

$$T_{ik} + t_{ij} - T_{jk} \leq (1 - x_{ijk})M \qquad \forall k \in K, (i, j) \in A \tag{1g}$$

$$T_{0k} \geq r_i \sum_{j \in \delta^+(i)} x_{ijk} \qquad \forall k \in K, i \in N \tag{1h}$$

$$T_{ik} \leq d_i \qquad \forall k \in K, i \in V \tag{1i}$$

A função objetivo (1a) minimiza o custo total de transporte. As restrições do problema garantem que teremos uma solução para o VRPRDD. A restrição (1b) garante que existirá somente uma rota saindo de i. As restrições de (1c) até (1e) são definidas coletivamente como restrições de fluxo que garantem que teremos caminhos saindo do depósito (0) e voltando ao depósito $(n + 1)$. Na restrição (1f), a capacidade máxima de cada rota k deve ser respeitada. A restrição de tempo expressa pela equação (1g) assegura que as variáveis de tempo estejam devidamente sincronizadas. O lado direito $(1 - x_{ijk})M$ é uma restrição *big-M*, onde M é uma constante grande que garante que a restrição seja imposta apenas quando $x_{ijk} = 0$. A restrição (1h) garante que todas as rotas só serão feitas depois que todos os pacotes estiverem disponíveis e a restrição (1i) cuida do *deadline* associado aos clientes.

2.3. Modelagem no VRPSolver para resolver o VRPRDD

Nesta seção, será mostrada uma modelagem utilizando o VRPSolver para o VRPRDD. A ideia do VRPSolver ([Pessoa et al. 2020]) é definir um modelo genérico tal que, quando um problema de otimização pode ser ajustado a esse modelo genérico, ele pode ser resolvido por meio de um algoritmo BCP, onde todos os subproblemas são modelados como um Problema de Caminho Mínimo com Restrições de Recursos (RCSPP). Para resolver os subproblemas, eles precisam ser modelados como grafos com recursos associados as arestas (ou vértices), além disso, uma formulação mestre precisa ser definida contendo

a função objetivo que irá ser minimizada e uma função de mapeamento que mapeia as variáveis de decisão da formulação aos grafos definidos. Segue a modelagem feita utilizando o Solver.

Primeiramente são definidos os grafos para a resolução do RCSPP, é construído um grafo $G^w = (V^w, A^w)$ para cada uma das W *release dates* diferentes na instância de entrada. Cada grafo correspondente a uma *release date* r_w é composto pelo conjunto de vértices representando os clientes que podem ser servidos partindo do depósito no momento r_w. Assim o conjunto de vértices pode ser definido como $V^w = \{v_i^w : r_i \leq r_w, r_w + t_{0i} \leq d_i\} \cup \{v_0^w\}$, onde o vértice v_0^w corresponde ao depósito. O conjunto de arestas liga todos os vértices de V^w, $A^w = \{(v_i^w, v_j^w) : v_i^w \in V^w, v_j^w \in V^w\}$. Os caminhos a serem construídos em cada um dos grafos devem sair e voltar para o depósito de modo que $v_{source}^w = v_{sink}^w = v_0^w$. Em cada um dos grafos são associados dois recursos, $R^k = R_M^k = \{1, 2\}$, o recurso de número 1 representa a capacidade, assim um caminho ao passar por uma aresta $a = (i, j) \in A^w$ consome $s_{a,1} = q_j$ do recurso 1, a restrição associada ao recurso 1 é que o consumo acumulado do recurso 1 ao passar por um vértice $v_i^w \in V^w$ deve estar no intervalo $[l_{v_i^w,1}, u_{v_i^w,1}] = [0, Q]$. O recurso de número 2 representa as restrições de tempo para que a entrega seja realizada, o consumo ao passar pela aresta $a = (i, j) \in A^w$ é dado por $s_{a,2} = t_{ij}$, e um caminho ao passar por um vértice $v_i^w \in V^w$ o consumo acumulado do recurso 2 deve estar no intervalo $[l_{v_i^w,2}, u_{v_i^w,2}] = [r_w + t_{0i}, d_i]$.

Para a formulação mestre será utilizado a variável binária de decisão x_{ij} que assumirá o valor 1 caso a aresta $(i, j) \in A$ seja utilizada na solução, e 0 caso contrário. A seguir a formulação mestre contendo a função objetivo (2a) que minimiza o custo das arestas utilizadas. E a restrição (2b) que garante que cada vértice será atendido uma única vez.

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2a}$$

$$\text{Sujeito a} \quad \sum_{(i,j)\in \delta^-(j)} x_{ij} = 1 \qquad \forall j \in N \tag{2b}$$

Por fim, a função objetivo é mapeada para os grafos com restrição de recurso por meio da definição da seguinte função, na qual cada variável (x_{ij}) é associada a todas as suas respectivas aparições em todos os W grafos, $M(x_{ij}) = \{(v_i^w, v_j^w)\}, v_i^w \in V^w, v_j^w \in V^w, w \in W$. Assim, cada caminho selecionado em um dos grafos afeta a função objetivo da formulação mestre. Em cada um dos grafos podem ser construídos até $|K|$ caminhos, mais formalmente definindo um limite superior e inferior $L^w = 0, U^w = |K|$, para cada $w \in W$.

2.4. Experimentos

Realizaram-se experimentos preliminares na modelagem proposta usando instâncias da VRPLIB modificadas com colunas de "release date" e "deadline". No entanto, são necessários experimentos e análises mais detalhadas antes da publicação do trabalho. As Tabelas 1 e 2 apresentam resultados para instâncias com janelas de tempo fixas e aleatórias, respectivamente, indicando um tempo de execução maior para instâncias com janelas de tempo aleatórias. Já nas Tabelas 3 e 4, que analisaram instâncias com janelas de tempo aleatórias e *release dates* discretizados em intervalos de 10 e 1 minutos, respectivamente. Naturalmente, devido a construção de um grafo para cada *release date*, tem-se tempos de execução piores para as instâncias com intervalos de 1 minuto.

Tabela 1. Resultados para instâncias com Janelas Aleatórias

Instância	Tempo (s)	Custo
F2h-n32	0.9	2227.5
F2h-n44	1.15	2131.5
F2h-n55	1.29	2259.9
F2h-n66	1.77	2960.4
F2h-n78	1.93	2865.2
F2h-n101	34.05	1131.0
F2h-n135	3319.78	2776.7

Tabela 2. Resultados para instâncias com Janelas Aleatórias

Instância	Tempo (s)	Custo
R-n32	1.72	1577.6
R-n44	9.74	1542.2
R-n55	7.48	1947.0
R-n66	62.55	2178.5
R-n78	56.26	2262.99
R-n101	232.46	1279.39
R-n135	4747.04	2556.1

Tabela 3. Resultados para instâncias com *release dates* discretizados de 10 em 10 minutos

Instância	Tempo (s)	Custo
Rr10-n32	1.78	1898.8
Rr10-n44	4.14	1466.3
Rr10-n55	5.74	2047.8
Rr10-n66	41.56	2629.2
Rr10-n78	313.21	1946.2
Rr10-n101	250.78	1333.19
Rr10-n135	> 7700	-

Tabela 4. Resultados para instâncias com *release dates* discretizados de 1 em 1 minuto

Instância	Tempo (s)	Custo
Rr1-n32	2.63	1627.1
Rr1-n44	3.51	1881.8
Rr1-n55	11.41	1792.69
Rr1-n66	254.25	2163.79
Rr1-n78	282.61	2983.0
Rr1-n101	1863.69	1296.3
Rr1-n135	> 7700	-

3. Resolvendo Instâncias Urbanas de Cidades Brasileiras

Nesta parte da pesquisa, o problema de entregas urbanas em grandes cidades brasileiras foi estudado. Para isso, foram utilizadas instâncias baseadas em entregas reais, obtidas através do *Benchmark* de Entregas Urbanas da Loggi (LoggiBUD). A fim de buscar soluções para esse desafio logístico, foi empregado um algoritmo BCP, também implementado pela ferramenta *VRPSolver*.

Uma das contribuições do trabalho foi a proposta de incorporar janelas de tempo em instâncias de entregas reais, levando em consideração as distâncias reais das ruas nas cidades brasileiras. Contudo, foi constatada a limitação de encontrar soluções exatas para essas instâncias devido à complexidade do problema. O projeto contou com o apoio do programa de bolsa de pesquisa da empresa Loggi (PBP Loggi).

3.1. Problema de Roteamento de Veículos com Janelas de Tempo - VRPTW

O Problema de Roteamento de Veículos com Janelas de Tempo (em ingles, *Vehicle Routing Problem* - VRPTW) é a variação do VRP mais discutida na literatura. Formalmente, definindo o VRPTW básico, dado um grafo direcionado completo $G = (V, A)$, onde o conjunto dos vértices V é composto pelo vértice 0, que representa o depósito, e o conjunto de vértices $N = \{1, ..., |N|\}$, que representa os clientes que serão visitados. A é o conjunto de arestas que liga cada par $(i, j) \in V \times V$, onde $i \neq j$. Associado a cada aresta existe um custo de viajem c_{ij} e um tempo de viagem $t_{ij} > 0$, onde t_{ij} inclui o tempo de serviço no vértice i. Em cada vértice $i \in V$ é associado uma demanda q_i e uma janela de tempo $[e_i, d_i]$, onde e_i e d_i representam respectivamente o primeiro e o último momento onde é possível visitar o vértice i. Para realizar as entregas são disponibilizados um frota de denotado por $K = \{1, .., |K|\}$. Os veículos são homogêneos e possuem capacidade Q.

O objetivo é minimizar o custo total das rotas tais que essas rotas obedeçam as restrições de capacidade e das janelas de tempo.

3.2. Método exato BCP usando VRPsolver

Utilizou-se o VRPSolver (Seção 2.3) para criar um modelo robusto e simplificado para avaliar seu desempenho em instâncias de cidades brasileiras. A formulação do VRPTW não foi apresentada por falta de espaço e não ser resultado direto da pesquisa, mas uma formulação semelhante para o VRPRDD pode ser encontrada na Seção 2.

3.3. Instâncias Adaptadas e Experimentos Computacionais

O VRPSolver foi utilizado para testar o modelo em dois conjuntos distintos de instâncias. O primeiro conjunto consiste nas instâncias clássicas de Solomon [Solomon 1987], enquanto o segundo é composto por instâncias adaptadas a partir do *loggiBUD*. Foram adaptadas 60 instâncias do *benchmark*, que incluíram janelas de tempo e tempo de serviço para cada cliente. A escolha dessas instâncias foi baseada na representatividade dos locais disponíveis. O *benchmark* é interessante porque inclui cidades brasileiras de dois estados diferentes (Pará - PA e Rio de Janeiro-RJ) e do Distrito Federal (DF), com demandas específicas para cada região.

Foram criados quatro tipos de intervalos para as janelas de tempo, incluindo instâncias com janelas de tempo aleatórias (R), instâncias com janelas de tempo fixas de 30 minutos (F30m), instâncias com janelas de tempo fixas de 1 hora (F1h) e instâncias com janelas de tempo fixas de 2 horas (F2h).

O método BCP do VRPSolver é mais eficiente para o VRPTW com janelas de tempo apertadas. Isso é evidenciado pelas instâncias usadas, onde nenhuma instância com janelas aleatórias pôde ser resolvida em menos de 2 horas, enquanto algumas instâncias com janelas fixas no estado do Pará (Figura 1) foram resolvidas em menos de 1,5 horas. Para obter mais informações sobre o processo de geração das instâncias e a experimentação, consulte [Clementino et al. 2022b].

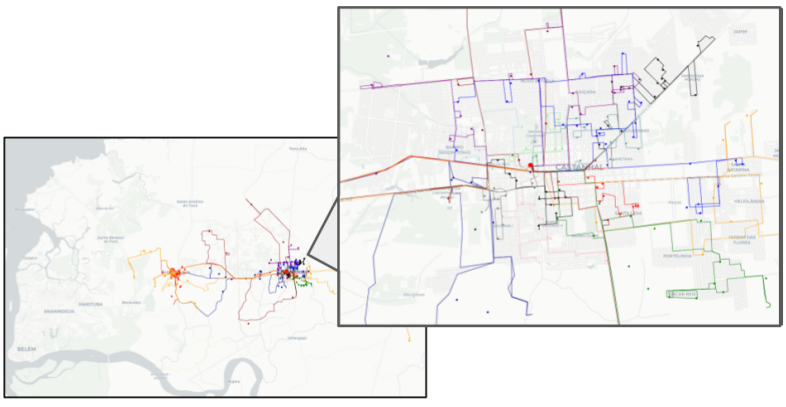

Figura 1. Solução para instância em Castanhal (PA). Fonte: [Loggi 2021].

4. Algoritmos e complexidade em grafos para instâncias especiais de VRPs

Durante o desenvolvimento desta pesquisa de iniciação científica, foram explorados diversos problemas de roteamento em teoria dos grafos, com foco especial na complexidade

de VRPs. Para isso, consideramos algumas classes especiais de grafos como instâncias de entrada. Veja nas Figuras 2 e 3 algumas dessas classes especiais de grafos estudadas.

Ao longo da pesquisa, foram abordados problemas como CVRP, SDVRP, TSP-rd e UVRP-rd, todos com o objetivo de investigar as complexidades envolvidas no roteamento em grafos. Ao explorar as particularidades desses problemas, foi possível identificar diversos desafios e oportunidades para o desenvolvimento de soluções mais eficientes e precisas em diferentes contextos aplicados.

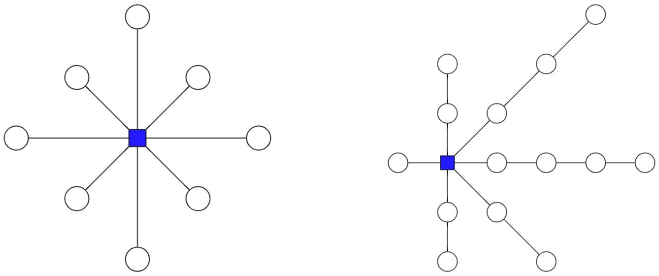

Figura 2. Grafos Estrela e Estrela Subdividida

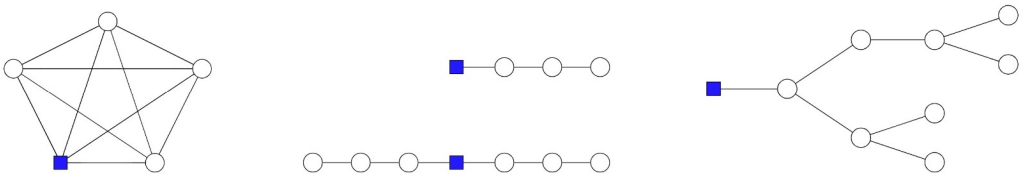

Figura 3. Grafos Completo, Caminho e Árvore

Além da revisão e compilação dos trabalhos da literatura, as maiores contribuições desse trabalho foram para os problemas com restrição de *release dates*. Para o TSP-rd, foram estendidos resultados anteriores vistos em [Reyes et al. 2018], onde o problema é resolvido em $O(n^2)$ para um grafo caminho onde o depósito se encontra na extremidade. E para o UVRP-rd, foi melhorado um algoritmo proposto em [Archetti et al. 2015] também para grafos caminhos. As subseções a seguir abordarão essas contribuições.

4.1. Problema do Caixeiro Viajante com *release dates* (TSP-rd)

Os resultados do TSP-rd em diferentes classes de grafos são apresentados na Tabela 5, destacando as contribuições em azul e as classes com problemas em aberto marcadas com dois pontos de interrogação. A pesquisa estendeu a programação dinâmica para TSP-rd em um grafo caminho com depósito em uma extremidade e desenvolveu relações de recorrência para minimizar o tempo de conclusão das rotas com um *deadline* para cada cliente.

Tabela 5. Complexidades para o TSP-rd em diferentes classes de grafos.

Classe do Grafo	time	distance	time w/deadline
Caminho (extremo)	$O(n^2)$	$O(n^2)$	$O(n^2)$
Caminho	$O(n^3)$	$O(n^3)$	$O(n^3)$
Estrela	$O(n \log n)$	$O(n)$	??
Estrela Generalizada	$O(n^{(m+1)})$	$O(n^{(m+1)})$	$O(n^{(m+1)})$

Assim como em [Reyes et al. 2018], é considerado que $r_i < r_{i+1}$ para todo $i < n$. A ideia é que se vários pedidos ficarem disponíveis simultaneamente, apenas o pedido

mais distante precisa ser considerado, uma vez que os demais pedidos podem ser entregues sem custo adicionais à solução.

Além disso, [Reyes et al. 2018] define *Rotas Não Entrelaçadas* como duas rotas K_1 e K_2 com $min\{i \mid i \in K_1\} < min\{j \mid j \in K_2\}$ são não entrelaçadas se e somente se $max\{r_i \mid i \in K_1\} < min\{r_j \mid j \in K_2\}$. Além disso é mostrado que sempre existe um cronograma de entregas contendo somente rotas não entrelaçadas que apresentam o custo mínimo c^* da solução ótima.

Foi desenvolvida uma relação de recorrência por [Reyes et al. 2018] para solucionar o problema, testando todas as rotas não entrelaçadas possíveis em um grafo caminho com o depósito no extremo. O algoritmo foi estendido para um grafo caminho qualquer com um conjunto de clientes a direita e esquerda do depósito, representados por N_r e N_l, respectivamente. A seguir, apresenta-se a relação para implementar a extensão.

- Estados: (i, j) para $i \in \{0, 1, 2, \ldots, n_l\}$ e $j \in \{0, 1, 2, \ldots, n_r\}$.
- Valores: $c(i, j)$ especificando o menor tempo de conclusão de um cronograma de entregas com rotas não entrelaçadas servindo os clientes $\{1, \ldots, i\} \subseteq N_l$ e $\{1, \ldots, j\} \subseteq N_r$ ou ∞ caso não seja possível servir os clientes $\{1, \ldots, i\} \subseteq N_l$ e $\{1, \ldots, j\} \subseteq N_r$ dentro do prazo final.

$$
\begin{aligned}
c(0,0) &= 0 \\
L(i,j) &= \min_{k \in \{0,\ldots,i-1\}} \{\max\{c(k,j), r_i^l\} + 2 \max_{k<p\leq i} \{d_p^l\} \mid \max\{c(k,j), r_i^l\} \leq \min_{k<p\leq i} \{l_p^l\}\} \\
R(i,j) &= \min_{w \in \{0,\ldots,j-1\}} \{\max\{c(i,w), r_j^r\} + 2 \max_{w<p\leq j} \{d_p^r\} \mid \max\{c(i,w), r_j^r\} \leq \min_{w<p\leq j} \{l_p^r\}\} \\
c(i,j) &= \min\{L(i,j), R(i,j)\}
\end{aligned}
$$

Para provar a corretude do algoritmo, considere o passo recursivo para computar $c(i, j)$. Nele é computado o custo de adicionar o cliente j a direita $R(i, j)$, ou o cliente i a esquerda $L(i, j)$ do depósito. Para a esquerda, o cliente i pode ser incluído em duas rotas, a primeira rota junto com os outros clientes $k + 1, \ldots, i - 1$, que é adicionada ao cronograma parcial que serve os clientes $\{1, \ldots, k\} \in N_l$ e $\{1, \ldots, j - 1\} \in N_r$ quando $k \leq i-2$. A segunda rota é a criação de uma nova rota contendo somente i quando $k = i-1$. Em ambos os casos a nova rota é viável se o tempo de despacho da rota $\max\{c(k, j), r_i^l\}$ não é maior que o menor prazo final de despacho da rota $\min_{k<p\leq i}\{l_p^l\}$. E o menor tempo de retorno ao depósito dessa rota possível é o tempo de despacho somado ao tempo de viagem da nova rota que é definido por $2 \max_{k<p\leq i}\{d_p^l\}$. De forma análoga acontece para a direita.

É possível pré-calcular em $O(n^2)$ os valores de máximo e mínimo em range para cada lado antes de utilizar a relação de recorrência e acessá-los em $O(1)$. Assim, cada um dos $n_r * n_l$ estados possíveis dessa recursão pode ser computado em $O(n)$. Por isso a complexidade final para encontrar o valor de $c(n_l, n_r)$ é $O(n^3)$.

4.2. Problema de roteamento de veículos não capacitados com *release dates* (UVRP-rd

O resumo dos resultados para esse problema está na Tabela 6, incluindo as contribuições deste trabalho em azul.

Tabela 6. Complexidades para o UVRP-rd em diferentes classes de grafos.

Classe do Grafo	distance	distance w/deadline
Caminho	$O(n^2)$ $O(n \log n)$	$O(n^3)$
Estrela	$O(n)$	$O(n)$

Assim como na Seção 4.1 será considerado que $r_i < r_{i+1}$ para todo $i < n$. Além disso, a proposição a seguir garante que as soluções para o UVRP-rd também são formadas por rotas não entrelaçadas. Uma solução ótima para o UVRP-rd em Caminho existe de modo que cada veículo execute uma rota na qual visite apenas clientes a esquerda N_l ou a direita do depósito N_r. Além disso o tempo de despacho da rota, diferentemente do TSP-rd são independentes. Por isso, o problema pode ser decomposto em dois subproblemas. Um a direita e outro a esquerda.

O Algoritmo 1, uma adaptação do algoritmo descrito em [Archetti et al. 2015], resolve o problema dos clientes à direita do depósito, mas pode ser usado de forma análoga para os clientes à esquerda. Ele emprega $\mathcal{R}$ para representar o conjunto de rotas, k para indexar o cliente mais distante do depósito que ainda não foi atendido pelas rotas em $\mathcal{R}$, e R_u para representar o conjunto de clientes atendidos pela rota atual. Com uma complexidade $O(n^2)$, o algoritmo é capaz de criar um grande número de rotas, possivelmente tão grande quanto o número de clientes, e explorar a possibilidade de incluir os clientes remanescentes que ainda não foram atendidos em cada nova rota.

Algoritmo 1: Solução para o UVRP-rd em um grafo Caminho

$\mathcal{R} \leftarrow \emptyset$, $k \leftarrow 1$, $R_u \leftarrow \emptyset$
repita
  Inserir em R_u todos os clientes i tal que $r_i \leq T - 2|d_k|$
  $\mathcal{R} \leftarrow \mathcal{R} \cup R_u$
  $R_u \leftarrow \emptyset$
  $k = \text{argmin}\{i \mid r_i > T - 2|d_k|\}$
até *que todos os clientes sejam servidos.*

Algoritmo 2: Solução para o UVRP-rd em um grafo Caminho em $O(n \log n)$

$\mathcal{R} \leftarrow \emptyset$, $k \leftarrow 1$, $R_u \leftarrow \emptyset$, $K \leftarrow \emptyset$
repita
  $k \leftarrow \text{argmin}\{i \mid r_i > T - 2|d_k| \text{ and } i \leq k\}$
  $K \leftarrow K \cup \{k\}$
até $k == n_r + 1$
$j \leftarrow 1$
para $i \in \{1, \ldots, |K|\}$ **faça**
  para $l \in \{j, \ldots, K_i - 1\}$ **faça**
    $R_u \leftarrow R_u \cup \{l\}$
  fim
  $j \leftarrow K[i]$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_u\}$
  $R_u \leftarrow \emptyset$
fim

Embora o Algoritmo 1 ofereça uma solução ótima, ele não leva em conta que as *release dates* estão ordenadas inversamente à distância em relação ao depósito. Isso

permite a eliminação da verificação de todos os clientes não atendidos, resultando em uma busca binária que encontra apenas o menor índice em que a data de lançamento é superior a um valor constante. O Algoritmo 2 é uma versão modificada do Algoritmo 1 que inclui um conjunto K, onde K_i representa o primeiro cliente a não ser atendido pela $i-$ésima rota. Com essa alteração o algoritmo passa a ser $O(n \log n)$.

5. Conclusão

Em conclusão, este artigo apresentou algumas contribuições em relação ao estudo de VRPs com restrições temporais. Foi abordado o desafio de garantir entregas eficientes dentro de janelas de tempo específicas, destacando a importância desse tipo de problema em serviços de entrega. Foram apresentadas soluções para instâncias do VRPTW baseadas em cidades brasileiras e do VRPRDD, bem como estudos em variações do VRP para algumas classes especiais de grafos. Essas contribuições fornecem ideias para avançar em direção a soluções mais eficientes e eficazes para VRPs com restrições temporais, e podem ser aplicadas em contextos práticos de logística e transporte.

Referências

Archetti, C., Feillet, D., and Speranza, M. G. (2015). Complexity of routing problems with release dates. *European journal of operational research*, 247(3):797–803.

Clementino, T., Rosas, J., de Freitas, R., and Uchoa, E. (2022a). Contribuições na logística de entrega urbana expressa de última milha usando grandes instâncias reais de cidades brasileiras. In *Anais do III Workshop Brasileiro de Cidades Inteligentes*, pages 1–12. SBC.

Clementino, T., Rosas, J., De Freitas, R., and Uchoa, E. (2022b). Solving real urban vrptw instances by applying a branch-cut-and-price via vrpsolver. In *2022 XVLIII Latin American Computer Conference (CLEI)*, pages 1–8. IEEE.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). *Chapter 5: The Vehicle Routing Problem with Time Windows*, pages 119–159.

Loggi (2021). loggibud: Loggi benchmark for urban deliveries. `https://github.com/loggi/loggibud`.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183(1):483–523.

Reyes, D., Erera, A. L., and Savelsbergh, M. W. (2018). Complexity of routing problems with release dates and deadlines. *European journal of operational research*, 266(1):29–34.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

Sun, X., Li, K., and Li, W. (2022). The vehicle routing problem with release dates and flexible time windows. *Engineering Optimization*, 54(12):2123–2139.

Yang, W., Ke, L., Wang, D. Z., and Lam, J. S. L. (2021). A branch-price-and-cut algorithm for the vehicle routing problem with release and due dates. *Transportation Research Part E: Logistics and Transportation Review*, 145:102167.