LIII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2021

Um Algoritmo Híbrido para o Problema de Roteamento de Veículos com Janelas de Tempo Robusto

Carlos Neves, Anand Subramanian
Universidade Federal da Paraíba – Centro de Informática
Rua dos Escoteiros, s/n – Mangabeira, João Pessoa – PB, 58055-000
carlosneves@eng.ci.ufpb.br, anand@ci.ufpb.br

Pedro Munari
Universidade Federal de São Carlos – Departamento de Engenharia de Produção
Rod. Washington Luís km 235 – SP-310 – São Carlos – SP, 13565-905
munari@dep.ufscar.br

RESUMO

Este trabalho aborda o problema de roteamento de veículos com restrições de janela de tempo, envolvendo demandas e tempos de viagem incertos. O problema foi abordado por meio do paradigma de otimização robusta, tratando-se as demandas e tempos de viagem como variáveis aleatórias pertencentes a conjuntos de incertezas com *budget*. Para resolver o problema, um algoritmo híbrido foi proposto. O algoritmo, que combina o *framework* de Busca Local Iterada com o método de particionamento de conjuntos, foi testado em instâncias geradas a partir do conjunto de *benchmark* de Solomon. Os testes computacionais demonstram que o algoritmo foi capaz não apenas de alcançar a maioria dos melhores limitantes superiores conhecidos, como de melhorar os resultados de várias das instâncias em aberto.

PALAVRAS CHAVE. roteamento de veículos, incertezas, otimização robusta, matheuristica

Tópicos: MH – Metaheurísticas, L&T – Logística e Transportes

ABSTRACT

This paper addressees the vehicle routing problem with time windows under uncertain demands and travel times. We treat it as a robust optimization problem, assuming the customer demands and travel times are random variables that belong to a budgeted uncertainty set. We propose a matheuristic that combines the Iterated Local Search framework with a set partitioning method. Computational experiments with instances generated from the Solomon benchmark set show the effectiveness of the method, as it was able to find the best known upper bounds for most of the instances, as well as improving the results of many of the open instances.

KEYWORDS. vehicle routing, uncertainties, robust optimization, matheuristic

Paper topics: MH – Metaheuristics, L&T – Logistics and Transport

1. Introdução

Em um processo de tomada de decisão, é importante considerar as características mais relevantes da situação em estudo e abordar de forma apropriada as incertezas envolvidas e os riscos a elas associados. Além disso, deve se beneficiar se possível de todo o ferramental matemático e computacional disponível, que viabilize a obtenção de decisões eficazes e que proporcionem o melhor desempenho na prática. Dessa forma, o desenvolvimento de modelos matemáticos e algoritmos que auxiliam nesse processo tem grande relevância prática, podendo trazer melhorias significativas para a tomada de decisão [Shapiro, 2007; Gendreau et al., 2014].

Ignorar as incertezas inerentes à situação de interesse, bem como os riscos envolvidos, pode levar ao fracasso de um processo de tomada de decisão. Além de ineficaz, uma decisão ruim pode resultar em perdas significativas e irreversíveis. Por outro lado, considerar as incertezas e riscos de forma efetiva é, em geral, um grande desafio em um processo de tomada de decisão. Além da dificuldade de se prever as fontes de incerteza, há também o desafio em se quantificar os possíveis cenários, bem como suas chances e riscos associados [Ben-Tal et al., 2009].

Nesse trabalho, aborda-se a tomada de decisão sob incertezas no contexto de roteamento de veículos, uma das atividades de maior relevância em operações logísticas. De modo geral, o roteamento de veículos consiste em determinar rotas eficientes para a entrega e/ou coleta de produtos, partindo de um ou mais pontos de origem até o destino final desses produtos, envolvendo clientes distribuídos geograficamente. O roteamento aparece também no transporte de passageiros, sendo fundamental na definição de rotas para transporte aéreo e rodoviário, bem como na definição de rotas para a prestação de serviços [Toth e Vigo, 2014]. Essa relevância prática, associada ao desafio computacional em se obter soluções ótimas ou de boa qualidade para exemplares de grande porte, faz com que o problema de roteamento de veículos (PRV) seja um dos mais estudados na literatura de Pesquisa Operacional (PO). Em particular, considera-se neste trabalho que os clientes possuem janelas de tempo que restringem os horários de visita dos veículos a intervalos específicos de tempo. Essa é uma variante bastante estudada na literatura, conhecida como PRV com janelas de tempo (PRVJT).

As incertezas mais comuns no roteamento de veículos são observadas em relação às demandas de clientes, tempos de viagem e tempos de serviço [Sungur et al., 2008; Munari et al., 2019]. O planejamento de rotas sem levar em consideração tais incertezas pode fazer com que, durante a operação, as rotas precisem ser modificadas de modo a contornar imprevistos, podendo comprometer o nível de serviço desejado ou aumentar consideravelmente o custo total da operação. Assim, o uso de modelos e algoritmos que permitam considerar essas incertezas para a determinação das rotas é de grande valia e pode trazer vantagem competitiva para as empresas.

Dentre as várias técnicas usadas em PO para lidar com incertezas no apoio à tomada de decisão, uma que tem sido usada com sucesso no contexto de roteamento de veículos é a Otimização Robusta (OR) [Ben-Tal e Nemirovski, 1999; Bertsimas e Sim, 2004]. Baseando-se em uma perspectiva de pior caso, a modelagem por meio da OR assume que as realizações dos parâmetros incertos pertencem a um conjunto de incertezas bem definido. Cada parâmetro é representado por um valor nominal, tipicamente seu valor esperado, e por uma variação de pior caso para esse valor nominal. Além disso, a variação total simultânea dos parâmetros deve pertencer ao conjunto de incertezas, que pode ser definido como uma intersecção de intervalos, um poliedro, um elipsoide, uma combinação de cenários obtidos a partir do histórico da operação, dentre outras formas. Pode-se inclusive utilizar uma distribuição de probabilidades nessa definição, embora uma das grandes vantagens da OR é não exigir tal uso. De fato, a escolha de uma distribuição de probabilidades para modelar as realizações dos parâmetros incertos é muitas vezes um inconveniente na prática.

Um dos conjuntos de incertezas mais usados para variantes do PRV é o conjunto intervalar com cardinalidade restrita originalmente proposto por Bertsimas e Sim [2003]. A restrição da cardinalidade é feita por meio de um parâmetro chamado *budget* de incerteza, que permite que o tomador de decisão controle o nível de incerteza de acordo com sua percepção. De forma simplicada, o *budget* controla o número máximo de parâmetros que podem assumir o pior caso em uma realização. Por exemplo, assumindo que as demandas dos clientes sejam incertas, um *budget* igual a 3 considera que no máximo três clientes em uma dada rota terão sua demanda atingindo o pior caso, enquanto as demandas dos demais clientes serão iguais a seus respectivos valores nominais. Incorporar um conjunto de incertezas desse tipo a modelos e algoritmos usados no planejamento das rotas pode trazer diversos benefícios, pois essas rotas ficam protegidas contra as variações consideradas, reduzindo significativamente o risco de falhas e a elevação de custos, conforme evidenciado nos resultados apresentados por Agra et al. [2013], Munari et al. [2019], entre outros.

Apesar do PRVJT determinístico ser uma das variantes mais estudadas na literatura, o número de trabalhos abordando o PRVJT robusto (PRVJTR) é relativamente pequeno. Embora mais realista que sua versão determinística, a resolução do PRVJTR se torna ainda mais desafiadora, conforme observado em trabalhos da literatura. Diferentes abordagens exatas têm sido propostas, envolvendo a proposição de modelos matemáticos, métodos *branch-and-cut* e métodos *branch-and-price* [Agra et al., 2012, 2013; Hu et al., 2018; Lu e Gzara, 2019; Munari et al., 2019]. Tais abordagens são eficazes na resolução de exemplares com até 50 clientes, mas apresentam dificuldade na resolução de exemplares de porte maior, podendo falhar na obtenção de soluções ótimas ou de boa qualidade em tempos computacionais razoáveis. Em relação a abordagens heurísticas, observam-se pouquíssimos trabalhos considerando o PRVJTR [Braaten et al., 2017; Hu et al., 2018]. Mesmo para a variante sem janelas de tempo, o PRV com restrições de capacidade robusto (PRVCR), o número de métodos heurísticos é bem reduzido [Gounaris et al., 2016; Subramanyam et al., 2020; Neves et al., 2020].

O objetivo deste trabalho é propor um método heurístico híbrido para o PRVJTR, considerando incertezas nas demandas e tempos de viagens. Além de reduzir a lacuna observada na literatura, pretende-se contribuir com o desenvolvimento de estratégias de solução que apoiem a tomada de decisão de forma eficiente em uma atividade tão relevante como a de roteamento de veículos. O método proposto é baseado na matheurística proposta por Neves et al. [2020] para o PRVCR, a qual é estendida para considerar janelas de tempo e incertezas nos tempos de viagem. Conforme apresentado nas próximas seções, essa extensão não é trivial, pois a análise de pior caso para a realização dos tempos de viagem exige atenção especial após a aplicação dos operadores de busca local. Diferentemente da análise de pior caso da demanda, os tempos de viagem que atingem o pior caso não são necessariamente aqueles com os maiores desvios, dado que as esperas ocasionadas pelas janelas de tempo podem absorver esses desvios. Além disso, estruturas e mecanismos de aceleração recentes na literatura foram adaptados ao PRVJTR.

O restante deste trabalho está organizado como segue. A Seção 2 apresenta uma descrição formal do problema. A Seção 3 descreve o algoritmo proposto. A Seção 4 apresenta os resultados computacionais. Por fim, a Seção 5 apresenta as considerações finais.

2. Descrição do problema

Seja $G = (V, A)$ um grafo direcionado em que V e A são os conjuntos de vértices e arcos, respectivamente. Tem-se que $V = V' \cup \{0\}$, onde o subconjunto $V' = \{1, \ldots, n\}$ representa os clientes e o vértice 0 é o depósito, e $A = \{(i, j) \mid i \neq j\}$. O custo de viajar por um arco $(i, j) \in A$ é c_{ij}, e a matriz de custos satisfaz a desigualdade triangular. Cada vértice $i \in V$ possui um tempo de serviço s_i, uma janela de tempo $[w_i^a, w_i^b]$ e uma demanda d_i, sendo $d_0 = 0$. Há também um tempo

de viagem t_{ij} associado a cada arco $(i, j) \in A$. Dada uma frota homogênea e ilimitada de veículos de capacidade C, o problema consiste em determinar um conjunto de rotas de custo mínimo de modo que: (*i*) cada rota deve começar no depósito, visitar um subconjunto dos clientes, e terminar no depósito; (*ii*) a capacidade do veículo não pode ser excedida; (*iii*) as janelas de tempo não devem ser violadas; (*iv*) todo cliente deve ser visitado exatamente uma vez, tendo sua demanda suprida.

Diferente da versão determinística do problema, no PRVJTR, tanto as demandas d_i quanto os tempos de viagem t_{ij} são tratados como parâmetros incertos e, assim, modelados como variáveis aleatórias. Dessa forma, seguindo o paradigma de Otimização Robusta [Ben-Tal e Nemirovski, 1999; Bertsimas e Sim, 2004], assume-se que as realizações dessas variáveis aleatórias pertençam a conjuntos de incertezas bem definidos, o que possibilita uma análise de pior caso das soluções. Nessa abordagem, uma solução só pode ser ótima se permanecer factível para todas as possíveis realizações do conjunto e, portanto, fica protegida contra incertezas.

Conjuntos de incerteza com cardinalidade restrita têm sido os mais utilizados na literatura do PRVJTR, dadas as suas vantagens teóricas e práticas [Agra et al., 2013; Munari et al., 2019; Lu e Gzara, 2019; De La Vega et al., 2020; Subramanyam et al., 2020]. A vantagem teórica está em seu caráter linear, que permite uma integração efetiva com modelos matemáticos e métodos de solução, seja por dualização ou pela linearização de equações recursivas. Já a vantagem prática é devido proporcionar uma interpretação intuitiva e interessante do impacto das incertezas para a solução, permitindo que o tomador de decisão controle o número máximo de parâmetros atingindo o pior caso em sua análise. Para esse controle, define-se um parâmetro chamado de *budget* de incerteza.

Seguindo a prática comum em trabalhos da literatura, definem-se aqui dois *budgets* de incerteza, a saber: Γ^d, número máximo de clientes cuja demanda assume seu pior caso; e Γ^t, número de máximo de arcos nos quais os tempos de viagem assumem seus piores casos. Assim, considera-se que $\bar{d}_i$ e $\bar{t}_{ij}$ são os valores nominais da demanda no nó $i \in V$ e do tempo de viagem do arco $(i, j) \in A$, respectivamente. Esses parâmetros atingem o pior caso quando acrescidos dos desvios associados $\hat{d}_i$ e $\hat{t}_{ij}$.

De modo a definir formalmente os conjuntos de incerteza relacionados à demanda dos clientes e aos tempos de viagem, sejam γ_i^d e γ_{ij}^t variáveis aleatórias primitivas com realizações no intervalo $[0, 1]$, de modo que $d_i = \bar{d}_i + \hat{d}_i\gamma_i^d$ e $t_{ij} = \bar{t}_{ij} + \hat{t}_{ij}\gamma_{ij}^t$. Assim, os conjuntos de incertezas referentes às demandas e aos tempos de viagem são dados respectivamente por:

$$U^d = \{d \in \mathbb{R}^{|V|} \mid d_i = \bar{d}_i + \hat{d}_i\gamma_i^d, \sum_{i \in V} \gamma_i^d \leq \Gamma^d, 0 \leq \gamma_i^d \leq 1, i \in V\}, \quad (1)$$

$$U^t = \{t \in \mathbb{R}^{|A|} \mid t_{ij} = \bar{t}_{ij} + \hat{t}_{ij}\gamma_{ij}^t, \sum_{(i,j) \in A} \gamma_{ij}^t \leq \Gamma^t, 0 \leq \gamma_{ij}^t \leq 1, (i, j) \in A\}. \quad (2)$$

Assim, para ser considerada (robusta) factível, uma solução deve satisfazer os itens (*i*) a (*iv*) definidos no início dessa seção para todas as possíveis realizações de demandas e tempos de viagem que pertençam aos conjuntos (1) e (2), respectivamente. Considerando S_r como o número de clientes em uma dada rota r, as restrições mencionadas devem ser satisfeitas considerando que, para cada rota: (i) quaisquer $\min\{S_r + 1, \Gamma^t\}$ arcos podem atingir o pior caso de tempo de viagem; (ii) quaisquer $\min\{S_r, \Gamma^d\}$ demandas podem atingir o pior caso. Fazendo-se $\Gamma^d = \Gamma^t = 0$, o problema se resume a sua versão determinística. Por outro lado, soluções associadas a valores mais altos de Γ^d e Γ^t são mais conservadoras, pois pressupõem um maior grau de incerteza quanto à possibilidade de desvios nas demandas e tempos de viagem.

3. Algoritmo proposto

O método proposto é uma *matheuristic* que une procedimentos de busca local com um método exato baseado em particionamento de conjuntos. O procedimento principal é apresentado no Algoritmo 1. Primeiramente, em I_{Max} iterações (linhas 4–9), uma solução construída de forma gulosa (linha 5, vide Subseção 3.1) é melhorada através do procedimento **ILS** (linha 7), que é baseado no *framework* da meta-heurística *iterated local search* (ILS) [Lourenço et al., 2003], e faz uso do procedimento RVND [Subramanian et al., 2010] para as buscas locais. Se a solução possui um custo melhor que f^*, s^* é atualizado (linhas 8–9). Após isso, s^* corresponde à melhor solução encontrada em I_{Max} iterações. Então, tenta-se obter uma solução melhor de forma exata através do procedimento **SetPartitioning** (linha 10, vide Subseção 3.5), que usa rotas que foram armazenadas em uma estrutura denominada *pool* no decorrer do algoritmo. Continua-se utilizando o procedimento **SetPartitioning** enquanto houver melhora (linhas 11–14), e a melhor solução encontrada é retornada (linha 15).

Algoritmo 1: ILS-SP

```
1  procedimento ILS-SP (s*, TempoMáximoSP)
2      f* ← ∞
3      PoolRotas ← {}
4      para i = 1, ..., I_Max faça
5          s ← Construção()
6          s' ← s
7          s' ← ILS(I_ILS, s, s', PoolRotas)
8          se f(s') < f* então
9              s*, f* ← s', f(s')
10     s' ← SetPartitioning(s*, PoolRotas, TempoMáximoSP)
11     faça
12         s* ← SetPartitioning(s', PoolRotas, TempoMáximoSP)
13         s', f* ← s*, f(s*)
14     enquanto f(s') < f*
15     retorna s*
```

3.1. Procedimento construtivo

O procedimento construtivo adotado baseia-se no método de inserção paralela proposto por Subramanian [2012] para a obtenção de uma soluções iniciais em problemas de roteamento de veículos. Nele, cada rota é primeiramente preenchida com um único cliente de forma aleatória. Os clientes restantes são então adicionados de forma também aleatória às rotas, com base em um critério de seleção que visa minimizar os custos de inserção. Durante o procedimento, as restrições de capacidade e janelas de tempo são relaxadas. Por isso, ao fim do procedimento, obtém-se uma solução possivelmente inviável tanto em termos de capacidade quanto de janelas de tempo, algo que é tratado como descrito na próxima seção.

3.2. Penalização de soluções inviáveis

O método utilizado para diferenciar soluções viáveis e inviáveis faz uso de penalizações, e estende o proposto por Neves et al. [2020], adicionando janelas de tempo. Para isso, considere s uma solução gerada pelo procedimento construtivo, e R_s seu conjunto de rotas. Além disso, seja c_r o custo de viagem de cada rota $r \in R_s$. A variável $w_{v_j,\gamma}$, que representa o tempo mínimo de chegada do veículo em cada cliente $v_j \in V'$ quando γ arcos atingem o pior caso de tempo de

viagem, pode ser calculada como:

$$w_{v_j,\gamma} = \begin{cases} w^a_{v_0}, & \text{se } j = 0; \\ \max\{w^a_{v_j}, w_{v_{j-1},\gamma}, +s_{v_{j-1}} + t_{v_{j-1},v_j}\}, & \text{se } \gamma = 0; \\ \max\{w^a_{v_j}, w_{v_{j-1},\gamma}, +s_{v_{j-1}} + t_{v_{j-1},v_j}, \\ w_{v_{j-1},\gamma-1} + s_{v_{j-1}} + t_{v_{j-1},v_j} + \hat{d}_{v_{j-1},v_j}\}, & \text{caso contrário.} \end{cases} \tag{3}$$

A demanda total da rota $d(r)$ no pior caso, por sua vez, pode ser obtida ordenando-se seus clientes em ordem não-crescente de $\hat{d}$ [Munari et al., 2019]. Com isso, pode-se calcular a violação das janelas de tempo e da capacidade na rota $r \in R_s$ a partir das equações (4) e (5):

$$V^d_r = \max\{0, d(r) - C\} \tag{4}$$

$$V^t_r = \sum_{v_j \in r} \max\{0, w_{v_j,\Gamma^t} - w^b_{v_j}.\} \tag{5}$$

Assim, e custo total de s pode então ser calculado como:

$$f(s) = \sum_{r \in R_s} (c_r + \beta(V^d_r + V^t_r)). \tag{6}$$

Na equação (6), β é um fator de penalização, alto o suficiente para que uma solução inviável tenha sempre um custo maior que qualquer solução viável.

3.3. Busca local

O procedimento de busca local é baseado no método RVND [Subramanian et al., 2010]. O método consiste em uma adaptação do procedimento VND [Mladenović e Hansen, 1997], que seleciona aleatoriamente dentre estruturas de vizinhança enquanto houver melhora. Os operadores usados nas buscas são recorrentes na literatura, e o conjunto de estruturas de vizinhança $\mathcal{N}$ é constituído por vizinhanças inter-rota ($\mathcal{N}_1, \ldots, \mathcal{N}_6$) e intra-rota ($\mathcal{N}_7, \ldots, \mathcal{N}_{11}$), descritas a seguir.

Shift ($\mathcal{N}_1, \mathcal{N}_2$). Uma subsequência de m clientes é retirada de uma rota r_1 e inserida em oura rota r_2, sendo $1 \leq m \leq 2$.

Swap ($\mathcal{N}_3, \ldots, \mathcal{N}_5$). Duas subsequências de tamanhos m e n, de rotas diferentes, são trocadas, sendo $1 \leq m, n \leq 2$.

2-OPT* ($\mathcal{N}_6$). Dois arcos (i_1, j_1) e (i_2, j_2) pertencentes a rotas diferentes, são removidos. A solução pode ser reconstruída de duas formas, ambas envolvendo a adição de dois novos arcos: (i) (i_1, i_2) e (j_1, j_2); (ii) (i_1, j_2) e (i_2, j_1).

Exchange ($\mathcal{N}_7$). Dois clientes de uma mesma rota têm suas posições trocadas.

Reinsertion ($\mathcal{N}_8, \ldots, \mathcal{N}_{10}$). Uma subsequência de m clientes é movida para outra posição na mesma rota, sendo $1 \leq m \leq 3$.

2-opt ($\mathcal{N}_{11}$). Dois arcos (i_1, j_1) e (i_2, j_2) são removidos de uma mesma rota, que é então reconstruída invertendo-se o segmento entre j_1 e i_2.

3.4. Perturbação

Foram utilizados dois tipos de perturbações distintos, escolhidos aleatoriamente com a mesma probabilidade. O primeiro deles consiste em trocar as posições de dois clientes aleatórios. O segundo consiste em trocar as rotas de dois clientes entre si, inserindo-os em posições aleatórias. Sempre que um dos métodos é escolhido, ele é repetido aleatoriamente entre 1 e 3 vezes.

3.5. *Set partitioning*

O método de particionamento de conjuntos busca consruir soluções utilizando-se rotas obtidas previamente, durante as buscas locais. Para isso, uma nova solução é construída de forma exata a partir de rotas armazenadas no decorrer do algoritmo. Isso é feito através da resolução de um MIP, formulado por meio de um conjunto de variáveis binárias λ, usadas para selecionar dentre as rotas armazenadas no conjunto R, como ilustrado a seguir.

$$\min \sum_{r \in R} c_r \lambda_r \tag{7}$$

$$\text{s.a} \sum_{r \in R_i} \lambda_r = 1 \qquad i \in V' \tag{8}$$

$$\lambda_r \in \{0, 1\} \qquad r \in R. \tag{9}$$

A função objetivo (7) consiste em minimizar o custo total das rotas. As restrições (8) garantem que cada cliente apareça em exatamente uma rota. Por fim, as restrições (9) definem o domínio das variáveis de decisão.

O Algoritmo 2 ilustra o procedimento SetPartitioning(). O procedimento gera um modelo matemático a partir das rotas armazenadas no *pool* (linhas 2–4). Após isso, um resolvedor retorna a melhor solução encontrada (linhas 5–6). Sempre que o resolvedor encontra uma solução, o procedimento IncumbentCallback() é chamado. Nele, caso a solução seja melhorada através do procedimento ILS(), atualiza-se o valor de Cutoff (linhas 8–12).

Algoritmo 2: *Set Partitioning*

```
1  procedimento SetPartitioning (s*, PoolRotas, TempoMáximoSP)
2      SPModel ← CriarModeloSP(PoolRotas)
3      MIPStart ← s*
4      Cutoff ← f(s*)
5      s* ← MIPSolver(SPModel, MIPStart, Cutoff, TempoMáximoSP, IncumbentCallback())
6      retorna s*
7  procedimento IncumbentCallback(Incumbent Solution)
8      s ← Incumbent Solution
9      s' ← s
10     s ← ILS(I_ILS, s, s', NULL)
11     se f(s) < f(s*) então
12         s*, Cutoff ← s, f(s)
```

3.6. Estruturas e mecanismos de aceleração

Os mecanismos e estruturas de aceleração utilizados, propostos por Malheiros et al. [2021], são descritos a seguir.

Memoization Move Descriptor (MMD). É uma extensão do método *Static Move Descriptor* [Zachariadis e Kiranoudis, 2010] [Beek et al., 2018]. Durante as buscas locais, o método armazena, para cada estrutura de vizinhança, uma descrição do melhor movimento encontrado para cada rota (ou par de rotas). Dessa forma, antes de explorar uma estrutura de vizinhança, verifica-se se a rota (ou par de rotas) já foi avaliada previamente. Se sim, verifica-se qual o melhor movimento (ou se a estrutura de vizinhança não melhora a solução). Caso contrário, a vizinhança é explorada completamente, e a descrição do melhor movimento é então armazenada.

Memoization Solution Descriptor (MSD). O método armazena ótimos locais já visitados pelo algoritmo. Caso o algoritmo encontre ótimos locais já visitados em iterações anteriores, a iteração corrente é abortada, evitando a realização de buscas em soluções que já foram exaustivamente perturbadas até que o critério de parada fosse atingido.

4. Resultados computacionais

O algoritmo foi implementado na linguagem C++ (g++ 9.3.0) e executado em um PC Intel® Core™ i7-2600 CPU 3,40GHz com 16 GB de memória RAM e sistema operacional Linux 64 bits. Os testes foram feitos no conjunto de instâncias concebido por Solomon [1987] para o PRVJT determinístico. Para adicionar incertezas, foi utilizado o método proposto por Munari et al. [2019]. O método consiste em tratar as demandas e tempos de viagem de cada instância como os valores nominais. Os desvios das demandas $\bar{d}_i$ e tempos de viagem $\bar{t}_{ij}$ são então calculados como $\hat{d}_i = \lfloor \alpha^d \times \bar{d}_i \rfloor$ e $\bar{t}_{ij} = 0{,}1 \times \lfloor 10 \times \alpha^t \times \bar{t}_{ij} \rfloor$. Dessa forma, diferentes variantes de cada instância são geradas, com $\Gamma^d, \Gamma^t \in \{0, 1, 5, 10\}$ e $\alpha^d, \alpha^t \in \{0{,}1,\ 0{,}25,\ 0{,}5\}$. As Tabelas 1–3 sintetizam os resultados para todos os conjuntos de instâncias após 5 execuções do algoritmo com TempoMáximoSP $= 60$ segundos e maxIter $= 50$. As instâncias foram separadas por tipo, número de clientes e *budgets*. As colunas Custo e Tempo ilustram o custo e tempo médios para as soluções do subconjunto de instâncias correspondente. A coluna Gap apresenta a diferença relativa entre os custos das soluções encontradas pelo algoritmo e os melhores limitantes superiores obtidos pelo método baseado em *branch-price-and-cut* proposto por Munari et al. [2019].

Como esperado, de modo geral, maiores valores de *budget* impactam negativamente na qualidade (custo), por restringirem o espaço de soluções viáveis. Embora o mesmo possa ser dito com relação aos tempos, observou-se que maiores valores de Γ^t, especificamente, possuem muito mais impacto nesse quesito. Isso decorre, muito provavelmente, da alta demanda computacional

Tabela 1: Resultados para instâncias com 25 clientes.

		C1			R1			RC1		
Γ^d	Γ^t	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**
0	0	190,59	0,000	0,2	463,37	0,000	0,2	350,24	0,000	0,2
1	0	202,68	0,000	0,2	463,42	0,011	0,2	360,91	0,222	0,2
5	0	238,50	0,000	0,2	463,42	0,011	0,2	434,39	0,547	0,3
10	0	239,08	0,018	0,2	463,42	0,011	0,2	439,54	0,399	0,2
0	1	192,34	0,000	0,2	1058,55	0,001	0,2	728,99	0,207	0,3
0	5	195,83	0,000	0,3	1068,76	0,030	0,3	755,64	0,000	0,4
0	10	195,83	0,000	0,4	1068,99	0,024	0,3	764,80	0,015	0,5
1	1	202,87	0,004	0,2	1058,55	0,001	0,2	739,60	0,047	0,3
5	5	238,95	0,011	0,3	1068,70	0,017	0,3	815,18	0,188	0,4
10	10	239,73	0,025	0,5	1068,88	0,000	0,3	818,32	0,696	0,5
		C2			R2			RC2		
Γ^d	Γ^t	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**
0	0	214,52	0,033	0,2	382,27	0,031	0,1	319,27	0,000	0,1
1	0	214,51	0,027	0,2	382,15	0,002	0,1	319,28	0,000	0,1
5	0	214,47	0,011	0,2	382,15	0,002	0,1	319,28	0,000	0,1
10	0	214,45	0,000	0,2	382,15	0,000	0,1	319,28	0,000	0,1
0	1	214,52	0,005	0,2	383,86	0,000	0,1	319,55	0,000	0,2
0	5	214,57	0,000	0,3	384,85	0,046	0,2	319,72	0,000	0,2
0	10	214,58	0,005	0,4	384,96	0,019	0,2	319,88	0,000	0,3
1	1	214,54	0,016	0,2	383,87	0,004	0,2	319,55	0,000	0,1
5	5	214,58	0,005	0,3	384,82	0,005	0,2	319,72	0,000	0,2
10	10	214,59	0,011	0,4	384,95	0,017	0,3	319,88	0,000	0,3

Tabela 2: Resultados para instâncias com 50 clientes.

		C1			R1			RC1		
Γ^d	Γ^t	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**
0	0	361,69	0,000	0,9	766,25	0,019	1,6	730,83	0,055	1,2
1	0	407,05	0,121	1,2	766,16	0,003	1,6	809,55	-0,112	1,4
5	0	454,87	0,155	1,2	766,97	0,022	1,9	845,61	0,240	1,6
10	0	459,43	0,051	1,5	767,93	0,034	1,7	852,40	0,419	1,5
0	1	365,08	0,000	1,1	2491,83	0,018	1,8	1161,56	0,139	1,5
0	5	370,30	0,000	1,7	2514,28	0,039	2,7	1221,81	0,049	2,0
0	10	370,30	0,000	2,5	2516,89	0,020	3,5	1227,07	0,094	3,0
1	1	407,93	0,162	1,5	2491,92	0,020	2,0	1215,91	-0,069	1,5
5	5	455,11	0,136	2,2	2514,75	0,009	3,1	1267,92	0,255	2,4
10	10	460,16	0,062	3,4	2517,27	0,060	3,8	1270,90	0,126	3,3
		C2			R2			RC2		
Γ^d	Γ^t	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**
0	0	357,50	0,000	0,8	615,73	0,036	1,7	573,34	0,348	0,7
1	0	357,50	0,000	0,8	615,63	0,020	1,8	573,33	0,346	0,7
5	0	360,69	0,000	0,9	616,45	-0,005	2,0	573,66	0,406	0,7
10	0	367,92	-0,038	0,9	615,68	0,022	2,0	573,37	0,353	0,7
0	1	357,50	0,000	1,1	619,66	0,005	2,2	577,09	0,179	0,7
0	5	357,50	0,000	2,0	620,80	0,038	3,1	577,74	0,161	1,1
0	10	357,50	0,000	3,1	620,90	-0,072	4,3	578,44	0,297	1,5
1	1	357,50	0,000	1,1	619,62	-0,001	2,1	577,19	0,197	0,7
5	5	360,69	-0,018	2,0	620,76	0,034	3,2	578,14	0,244	1,1
10	10	368,02	-0,002	3,0	620,95	0,023	4,4	578,43	0,295	1,6

Tabela 3: Resultados para instâncias com 100 clientes.

		C1			R1			RC1		
Γ^d	Γ^t	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**
0	0	826,70	-0,001	3,9	1173,74	0,014	53,2	1334,51	0,002	49,5
1	0	924,33	0,041	8,2	1174,69	-0,002	57,1	1348,39	0,049	58,7
5	0	1042,57	-0,083	20,5	1180,97	-0,055	74,3	1386,50	-0,130	71,6
10	0	1056,39	-0,083	23,3	1186,65	-0,057	78,2	1402,85	0,068	81,2
0	1	846,73	-0,149	4,8	1154,17	-0,011	67,2	1400,48	0,074	50,6
0	5	862,05	-0,118	8,6	1195,33	0,008	91,1	1470,80	-0,126	62,7
0	10	862,40	-0,141	12,0	1203,42	0,013	129,1	1481,87	0,057	94,9
1	1	926,34	-0,076	11,2	1155,05	0,061	75,2	1412,70	0,067	61,4
5	5	1045,70	-0,108	28,2	1197,90	0,062	118,7	1492,02	-0,057	91,7
10	10	1059,81	-0,085	43,9	1206,99	-0,072	162,4	1506,26	0,001	115,7
		C2			R2			RC2		
Γ^d	Γ^t	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**	**Custo**	**Gap (%)**	**Tempo (s)**
0	0	587,38	0,000	12,1	873,35	0,085	44,5	1002,53	0,207	22,2
1	0	587,37	0,000	12,0	873,38	-1,369	43,8	1002,65	0,169	22,4
5	0	593,66	-0,126	12,5	873,32	-0,316	49,5	1002,48	0,153	26,7
10	0	613,40	-0,944	14,7	873,46	-1,334	55,0	1002,40	0,070	28,8
0	1	593,43	0,000	15,5	876,69	-1,110	53,8	1006,89	0,077	25,3
0	5	599,81	0,000	33,8	879,75	-1,647	87,8	1011,49	0,066	42,4
0	10	599,81	0,000	62,8	880,18	-1,814	127,5	1012,05	-0,042	62,3
1	1	593,43	0,000	15,2	876,41	-1,926	51,8	1006,80	0,086	27,5
5	5	604,78	-0,251	28,5	879,88	-2,067	90,2	1011,08	0,014	44,0
10	10	622,25	-1,459	44,7	880,49	-1,984	143,5	1012,05	-0,119	67,0

na avaliação de soluções imposta por esses valores durante as buscas locais. Também é possível notar que a distribuição geográfica dos clientes pode ser um fator limitante para o algoritmo, já que instâncias com prefixo R (distribuição geográfica aleatória parcial ou total dos clientes) apresentam *gaps* maiores no geral.

Contudo, o contrário é observado no conjunto nos conjuntos de instâncias com 100 clientes, que apresentam *gaps* médios negativos principalmente em instâncias com altos valores de Γ^d e Γ^t. Isso pode ser visto com ainda mais frequência no subconjunto de instâncias R2. Além de possuírem distribuições geográficas de clientes totalmente aleatórias, as instâncias desse subconjunto possuem janelas de tempo mais largas, sendo notoriamente mais difíceis para métodos exatos. Por esses motivos, o algoritmo foi capaz de obter melhores limitantes superiores para mais de 14% das instâncias com 100 clientes.

5. Considerações finais

Neste trabalho, foi proposta uma abordagem híbrida para resolver o PRVJTR, que une heurísticas de buscas locais a um método exato de particionamento de conjuntos. Embora tenha sido possível se aproveitar de muitos elementos de trabalhos anteriores, foi necessário desenvolver novos métodos para lidar com as incertezas quanto aos tempos de viagens. Além disso, estruturas e mecanismos de aceleração recentes na literatura foram adaptados ao problema, e então implementados no algoritmo. Os testes mostram que o algoritmo proposto é promissor, tendo obtido novos limitantes superiores para várias instâncias em aberto. Contudo, ainda há espaço para melhorias, dado que o algoritmo não foi capaz de alcançar os limitantes superiores conhecidos para algumas instâncias até o momento da escrita deste trabalho.

Como sugestão para trabalhos futuros, propõe-se a adição de novos mecanismos de aceleração ao método, como o *Feasible Range Search* (FRS) [Malheiros et al., 2021], bem como a calibração dos parâmetros do método através de ferramentas como o IRACE [López-Ibáñez et al., 2016]. Além disso, propõe-se a adaptação de outras heurísticas estado da arte na resolução de problemas clássicos como o PRVC e PRVJT para o PRVJTR.

Referências

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., e Requejo, C. (2013). The robust vehicle routing problem with time windows. *Computers & Operations Research*, 40(3): 856–866.

Agra, A., Christiansen, M., Figueiredo, R., Magnus Hvattum, L., Poss, M., e Requejo, C. (2012). *Layered Formulation for the Robust Vehicle Routing Problem with Time Windows*, p. 249–260. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-32147-4.

Beek, O., Raa, B., Dullaert, W., e Vigo, D. (2018). An efficient implementation of a static move descriptor-based local search heuristic. *Computers & Operations Research*, 94:1–10.

Ben-Tal, A. e Nemirovski, A. (1999). Robust solutions of uncertain linear programming. *Operations Research Letters*, 25:1–13.

Ben-Tal, A., El Ghaoui, L., e Nemirovski, A. (2009). *Robust optimization*. Princeton University Press.

Bertsimas, D. e Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71. ISSN 0025-5610.

Bertsimas, D. e Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

Braaten, S., Gjønnes, O., Hvattum, L. M., e Tirado, G. (2017). Heuristics for the robust vehicle routing problem with time windows. *Expert Systems with Applications*, 77:136–147.

De La Vega, J., Munari, P., e Morabito, R. (2020). Exact approaches to the robust vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research*, 124: 105062.

Gendreau, M., Jabali, O., e Rei, W. (2014). Stochastic vehicle routing problems. In *Vehicle routing: problems, methods, and applications, 2nd edn*, p. 213–239. Society for Industrial and Applied Mathematics.

Gounaris, C. E., Repoussis, P. P., Tarantilis, C. D., Wiesemann, W., e Floudas, C. A. (2016). An adaptive memory programming framework for the robust capacitated vehicle routing problem. *Transportation Science*, 50(4):1239–1260.

Hu, C., Lu, J., Liu, X., e Zhang, G. (2018). Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers & Operations Research*, 94:139–153.

Lourenço, H. R., Martin, O. C., e Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, p. 320–353. Springer.

Lu, D. e Gzara, F. (2019). The robust vehicle routing problem with time windows: Solution by branch and price and cut. *European Journal of Operational Research*, 275(3):925–938.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., e Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Malheiros, I., Ramalho, R., Passeti, B., Bulhões, T., e Subramanian, A. (2021). A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem. *Computers & Operations Research*, 129: 105196.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Munari, P., Moreno, A., De La Vega, J., Alem, D., Gondzio, J., e Morabito, R. (2019). The robust vehicle routing problem with time windows: Compact formulation and branch-price-and-cut method. *Transportation Science*, 53(4):1043–1066.

Neves, C., Subramanian, A., e Munari, P. (2020). Um algoritmo híbrido para o problema de roteamento de veículos robusto. *Anais do LII Simpósio Brasileiro de Pesquisa Operacional*.

Shapiro, J. F. (2007). *Modeling the supply chain*, volume 2. Thomson Brooks/Cole.

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

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., e Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899 – 1911. Metaheuristics for Logistics and Vehicle Routing.

Subramanian, A. (2012). *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. PhD thesis, Universidade Federal Fluminense.

Subramanyam, A., Repoussis, P. P., e Gounaris, C. E. (2020). Robust optimization of a broad class of heterogeneous vehicle routing problems under demand uncertainty. *INFORMS Journal on Computing*, 32(3):661–681.

Sungur, I., Ordonez, F., e Dessouky, M. (2008). A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions (Institute of Industrial Engineers)*, 40(5):509–523.

Toth, P. e Vigo, D. (2014). *Vehicle Routing: Problems, Methods and Applications*. MOS-SIAM Series in Optimization, Second edition.

Zachariadis, E. E. e Kiranoudis, C. T. (2010). A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. *Computers & Operations Research*, 37(12):2089–2105.