UM ALGORITMO EXATO PARA O PROBLEMA DA COBERTURA MULTI-VEÍCULO

Patrick Doglio (Universidade Federal Fluminense) patrickdoglio@id.uff.br
Marcos Costa Roboredo (Universidade Federal Fluminense) mcroboredo@id.uff.br

Resumo

Este trabalho estuda o Problema da Cobertura Multi-Veículo (PCMV). O PCMV considera um grafo completo $\boldsymbol{G} = (\boldsymbol{V} = \{\boldsymbol{0}\} \cup \boldsymbol{M} \cup \boldsymbol{O} \cup \boldsymbol{C}, \boldsymbol{E})$, onde 0 representa um depósito e os subconjuntos $\boldsymbol{M}$, $\boldsymbol{O}$ e $\boldsymbol{C}$ são chamados de conjunto de vértices mandatórios, opcionais e clientes, respectivamente. O PCMV consiste em definir rotas de custo total mínimo começando e terminando no depósito de modo que cada vértice mandatório é visitado por exatamente uma das rotas e, para cada vértice cliente, existe ao menos um vértice opcional visitado por uma das rotas tal que a distância entre estes dois vértices seja menor ou igual a um raio pré-estabelecido. Nós apresentamos um algoritmo *branch-and-cut-and-price* exato para o PCMV gerado a partir da proposição de um modelo na ferramenta VRPSolver, muito utilizada para problemas de roteamento em tempos recentes. Apresentamos ainda diversos experimentos computacionais onde os resultados mostram que o método proposto apresenta uma melhor performance computacional do que o melhor método exato da literatura, sendo capaz de resolver 5 instâncias que estavam em aberto até o momento.

Palavras-Chaves: Otimização Combinatória, Roteamento de Veículos, VRPSolver

1. Introdução

Nas últimas décadas, Problemas de Roteamento de Veículos (PRVs) tem sido amplamente estudados por um grande número de pesquisadores ao redor do mundo. O grande interesse por este tipo de problema é porque os PRVs podem modelar diversas situações da vida real, em especial aquelas que envolvendo distribuição física. O primeiro VRP foi definido em Dantzig e Ramser (1959) e consiste em definir rotas ótimas de entrega de um depósito para um dado número de clientes. De lá para cá, um vasto número de variantes tem sido propostas considerando diferentes condições tais como capacidades, janelas de tempo, frota heterogênea, múltiplos depósitos, coleta e entrega, etc.

Este trabalho aborda o Problema da Cobertura Multi-Veículo (PCMV). O PCMV considera um grafo completo $G = (V, E)$ onde V e E representam respectivamente o conjunto de vértices e arestas. Para cada aresta $e \in E$, é conhecido um custo c_e, que representa a custo de travessia da aresta. O conjunto de vértices é particionado em subconjuntos, $V = \{0\} \cup M \cup O \cup C$, onde o vértice 0 representa um depósito e os subconjuntos M, O e C são chamados de conjunto de vértices Mandatórios, Opcionais e Clientes, respectivamente. O objetivo do problema é definir um conjunto de rotas com custo total mínimo e respeitando as seguintes condições:

- Cada rota começa e termina no depósito;
- Todo vértice mandatório de M é visitado exatamente uma vez por uma única rota;
- Os vértices clientes de C não podem ser visitados diretamente por nenhuma rota;
- Para cliente $j \in C$, existe ao menos um ponto $i \in O$ tal que i é visitado por uma das rotas e $c_{(i,j)} \leq \beta$, onde $\beta > 0$ é um raio pré-estabelecido;
- Cada rota visita no máximo p pontos e tem custo máximo C_{max};

Dada a definição do PCMV, podemos observar que os clientes não são visitados diretamente, assim como acontece na grande maioria dos PRVs. Por este motivo, no PCMV, o cliente é dito estar coberto ao invés de atendido. Como já mencionado por Lopes (2012), o PCMV tem algumas aplicações práticas, sendo a mais comum na distribuição de unidades móveis de saúde.

O PCMV é conhecido em inglês como "Multi-Vehicle Covering Tour Problem" e foi proposto por Hachicha et al. (2000), onde heurísticas foram apresentadas para o problema. De lá para cá, devido a importância prática e teórica do PCMV, diversos trabalhos vem propondo algoritmos tanto exatos quanto heurísticos para a resolução deste problema, dentre os quais destacamos Naji-Azimi et al. (2012), Lopes et al. (2013), Ha et al. (2013), Jozefowiez (2014), Kammoun et al. (2015) e Glize et al. (2020).

Além da versão clássica do PCMV, alguns autores também tratam variantes do problema, como por exemplo, a variante com múltiplos depósitos proposta por Allahyari et al. (2015), a variante que visa minimizar a soma dos tempos de chegada nos vértices visitados proposta por Flores-Garza et al. (2017), a variante em que vértices clientes podem ter de ser cobertos mais de uma vez proposta por Pham et al. (2017), a variante que visa maximizar a soma das probabilidades dos clientes serem cobertos proposta por Karaouglan et al. (2018) e a variante com otimização de velocidade proposta por Margolis et al. (2021).

No melhor do nosso conhecimento, o melhor algoritmo exato para a versão clássica do PCMV foi proposto por Glize et al. (2020). Os autores propuseram um algoritmo *branch-and-cut-and-price* (BCP) para o problema capaz de resolver instâncias com até $|V| = 200$. Apesar dos bons resultados, diversas instâncias não foram resolvidas até a otimalidade.

Este trabalho propõe um novo algoritmo BCP exato o PCMV gerado a partir da proposição de um modelo na ferramenta VRPSolver proposta por Pessoa et al. (2020), que foi utilizada recentemente por Queiroga et al. (2020) e Damião et al. (2021) para outros PRVs. Para comprovar a robustez do método proposto neste trabalho, nós apresentamos diversos resultados computacionais, incluindo uma comparação com o melhor algoritmo exato proposto por Glize et al. (2020). Resultados mostram que o método proposto apresenta uma melhor performance computacional para a maior parte das instâncias, sendo capaz de comprovar a otimalidade de 5 instâncias que estavam em aberto até o momento na literatura.

O restante deste trabalho é organizado da seguinte forma. A Seção 2 apresenta uma revisão relativa à ferramenta VRPSolver. A Seção 3 apresenta o modelo VRPSolver proposto para o PCMV. A Seção 4 descreve diversos experimentos computacionais. Finalmente, a Seção 5 sumariza nossas conclusões e perspectivas de trabalhos futuros.

2. Revisando o modelo VRPSolver genérico

Nesta seção, nós apresentamos o modelo VRPSolver genérico, que contém uma formulação de Programação Linear Inteira Mista (PLIM) com variáveis representando rotas viáveis. O conjunto de rotas viáveis é modelado por caminhos com restrição de recursos em um ou diversos grafos direcionados. O VRPSolver resolve o modelo usando um algoritmo BCP genérico que gera as variáveis de caminho dinamicamente através da resolução de subproblemas de *pricing* modelados como Problema do Caminho Mais Curto com Restrições de Recursos. Além da formulação PLIM e dos grafos direcionados, o usuário pode definir os chamados *packing sets*, que permite a ativação de eficientes componentes muito utilizadas em algoritmos BCPs para PRVs, como por exemplo, relaxação *ng-path*, cortes de capacidade arredondados, cortes de rank-1 com memória limitada e enumeração de rotas.

Neste trabalho é apresentada uma descrição simplificada do VRPSolver que seja suficiente para que o modelo proposto possa ser entendido. Para uma descrição completa de todas as componentes da ferramenta, ver Pessoa et al. (2020).

2.1. Grafo para os subproblemas de *pricing*

Seja $G^1 = (V^1, A^1)$ um grafo com dois especiais vértices: v_{origem} e $v_{destino}$ que podem ser iguais ou diferentes. Associado a este grafo, existe um conjunto de recursos R, onde para cada recurso $r \in R$ e para cada arco $a \in A^1$, o consumo de r em a é denotado por q_{ar}. Para cada vértice $v \in V^1$ e cada recurso $r \in R$, existe um intervalo $[l_{vr}, u_{vr}]$ indicando que o consumo acumulado de r em um dado caminho iniciado em v_{origem} tem que estar dentro deste intervalo para que v possa ser visitado neste caminho.

2.2. Formulação PLIM

Seja P o conjunto de caminhos viáveis em G^1. O número de vezes que um dado arco a aparece em um dado caminho p é denotado por h_a^p. A formulação mestre usa variáveis inteiras $x_j, 1 \leq j \leq n$ e variáveis inteiras λ_p indicando o número de vezes que o caminho p é usado.

$$\min \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$s.a. \sum_{j=1}^{n} \alpha_{ij} x_j \geq d_i, \qquad i = 1, \dots, m \tag{2}$$

$$x_j = \sum_{p \in P} \left(\sum_{a \in M(x_j)} h_a^p \right) \lambda_p, \qquad j = 1, \dots, n \tag{3}$$

$$L \leq \sum_{p \in P} \lambda_p \leq U \tag{4}$$

$$\lambda_p \in \mathbb{Z}_+, \qquad p \in P \tag{5}$$

$$x_j \in \mathbb{Z} \qquad j = 1, \dots, n \tag{6}$$

A Formulação (1) – (5) é uma formulação bastante genérica, que é definida pelo usuário de acordo com o problema a ser resolvido. As restrições (3) garantem a relação entre as variáveis

x e λ. Cada uma destas equações usa um conjunto $M(x_j)$ representando o mapeamento da variável x_j. O mapeamento é definido pelo usuário e é tal que $M(x_j) \subseteq A^1$. A restrição (4) garante que a solução ótima possui no mínimo L caminhos no grafo G^1 e no máximo U, onde L e U são definidos pelo usuário.

Uma vez que o usuário definiu todas as componentes do grafo G^1 e todas as componentes da Formulação (1) – (5), as próximas etapas são realizadas pela ferramenta. O VRPSolver então resolve a formulação através do clássico algoritmo de geração de colunas considerando como formulação mestre como sendo a relaxação linear da formulação (1) – (5), mas apenas considerando as variáveis λ (as variáveis x são substituídas com base nas equações (3)). A formulação mestre é então resolvida através de um procedimento iterativo onde, em cada iteração, uma versão restrita da formulação mestre com um subconjunto das variáveis λ é resolvida e, então, variáveis λ com custo reduzido negativo são então incorporadas a formulação. Tais variáveis são encontradas através da resolução do subproblema de *pricing*, que consiste em encontrar um ou mais caminhos viáveis em G^1com custo reduzido negativo. Para isso, o VRPSolver utiliza um algoritmo chamado algoritmo de *labeling*. Para mais detalhes sobre este algoritmo, ver Pessoa et al. (2020).

O VRPSolver possui implementado um algoritmo BCP genérico que resolve a formulação mestre em cada nó da árvore de busca. *Branch* sobre variáveis x é utilizado para que a Formulação PLIM (1) – (5) seja resolvida até a otimalidade.

A definição da Formulação (1) – (5) e do grafo G^1 já são suficientes para o usuário obter um algoritmo BCP para um dado problema. No entanto, o usuário pode ainda definir o chamado conjunto de *packing sets* $\wp^v$, que corresponde a uma união de subconjuntos de V^1 de modo que, para cada um destes subconjuntos, no máximo um de seus elementos é visitado na solução ótima e no máximo uma única vez. A definição de *packing sets* é de extrema importância pois permite ao VRPSolver ativar algumas componentes avançadas para algoritmos BCP como já mencionado anteriormente.

3. Modelo VRPSolver para o PCMV

Nesta seção apresentamos o modelo VRPSolver para o PCMV. Tal modelo utiliza as constantes $ed(i,j)$ definidas como $ed(i,j) = (i,j)$, se $i < j$ e $ed(i,j) = (i,j)$, caso contrário. O modelo

considera ainda um único grafo $G^1 = (V^1, A^1)$ para o subproblema de *pricing* com $V^1 = \{v_0^1, \dots, v_{|M|+|O|}^1\}$ e $A^1 = \{(i,j),(j,i) | i,j \in V^1, i \neq j\}$. Os vértices origem e destino são $v_{origem} = v_{destino} = v_0^1$. Temos ainda o uso de dois recursos ($R^1 = \{r_1, r_2\}$), onde r_1 e r_2 são usados para garantir que cada rota visite no máximo p pontos e para garantir que cada rota tenha um custo máximo C_{max}, respectivamente. Assim, temos $[l_{ir_1}, u_{ir_1}] = [0, p]$ e $[l_{ir_2}, u_{ir_2}] = [0, C_{max}]$, para cada $i \in V^1$. Os consumos do recurso r_1 é dado da seguinte forma: $q_{(i,j)r_1} = 1$, para todo $a = (i,j) \in A^1$ com $j \neq v_0^1$. Para os demais arcos, $q_{ar_1} = 0$. As definições prévias de consumo do recurso r_1 garantem que toda vez que um vértice mandatório ou opcional for visitado, o consumo acumulado no caminho aumenta uma unidade e, como o limite superior de cada vértice i este recurso é $u_{ir_1} = p$, nunca teremos uma rota visitando mais do que p vértices. Já para o recurso r_2, definimos $q_{ar_2} = c_{ed(i,j)}$, para todo $a = (v_i^1, v_j^1) \in A^1$. Notemos que o consumo de cada arco é exatamente o custo de travessia associado e, como o limite superior de cada vértice i para este recurso é $u_{ir_2} = C_{max}$, nunca teremos uma rota com custo maior do que C_{max}. A formulação usa variáveis inteiras x_e, para cada $e \in E'$, onde $E' = \{e = (e_1, e_2) \in E | e_1, e_2 \notin C\}$. A formulação usa ainda as constantes $\delta(i)$ representando o conjunto de arestas em E' adjacentes a i e as constantes $\delta'(j)$, para todo vértice cliente $j \in C$, definidas como conjunto de vértices opcionais que estão a uma distância de no máximo β deste cliente. Em outras palavras, $\delta'(j) = \{i \in O | c_{ed(i,j)} \leq \beta\}$. A formulação segue:

$$\text{Min} \quad \sum_{e \in E'} c_e x_e \tag{7}$$

$$\sum_{e \in \delta(i)} x_e = 2, \qquad i \in M \tag{8}$$

$$\sum_{i \in \delta'(j)} \sum_{e \in \delta(i)} x_e \geq 2, \quad j \in C \tag{9}$$

A função objetivo (7) visa o minimizar o custo total das rotas. As restrições (8) garantem que cada vértice mandatório tem grau 2, ou seja, vai ser visitado exatamente uma vez. Já as restrições (9), garantem que, para cada cliente $j \in C$, ao menos um vértice opcional de $\delta'(j)$ terá grau 2. O mapeamento das variáveis x é definido como $M(x_e) = \{(i,j),(j,i)\}$, para casa $e = (i,j) \in E'$. $L^1 = 1$ e $U^1 = |M| + |O|$, indicando que temos no mínimo uma rota e, no pior dos casos, temos uma rota para cada vértice possível de ser visitado. Os *packing sets* são definidos como $\wp^v = \cup_{i \in M \cup O} \{v_i^1\}$, indicando que na solução ótima cada vértice mandatório ou opcional é visitado no máximo uma vez.

4. Experimentos computacionais

Nesta seção, nós apresentamos resultados computacionais para o PCMV. Todos os experimentos foram executados em um computador com um processador Intel Core i7-4790 com 3.6 GHz e 16 GB de memória RAM. O sistema operacional utilizado foi o Ubuntu 18.04.2 LTS. Cada instância é resolvida usando uma única *thread*. O CPLEX 12.9 é usado como resolvedor para programação linear e programação linear inteira mista.

As instâncias testadas são as mesmas usadas por Glize et al. (2020). Elas são geradas a partir de 6 conjuntos de dados do TSPLIB: KroA100, KroB100, KroC100, KroD100, KroA200, KroB200, sendo os 4 primeiros com 100 pontos e os 2 últimos com 200 pontos. Cada conjunto fornece as coordenadas de cada um dos pontos. A partir destas coordenadas, o custo de travessia cada aresta é então calculado como sendo a distância euclidiana entre os pontos. O valor de β é tal que cada vértice de O cobre pelo menos um vértice cliente de C e cada vértice cliente de C é coberto por pelo menos dois vértices de O.

As instâncias são nomeadas na forma $X - |M| - |O| - |C| - p$ onde X é formado por uma letra e um número. A letra pode ser A, B, C ou D representando se a instância é derivada da base de dados KroA, KroB, KroC ou KroD e o número pode ser 1 ou 2, representando se o conjunto da qual a instância é derivada possui 100 ou 200 pontos, respectivamente. Assim, por exemplo, a instância A2-10-89-100-4 é derivada do conjunto KroA200 com 1 depósito, 10 vértices mandatórios, 89 vértices opcionais, 100 vértices clientes e $p = 4$. Nós não consideramos o valor de C_{max} no nome das instâncias pois para todas as instâncias testadas, temos $C_{max} = +\infty$, seguindo a literatura.

Nós fizemos uma comparação do método proposto com o melhor método exato encontrado na literatura, proposto por Glize et al. (2020), com base nas mesmas 96 instâncias que este método testou. O processador usado pela literatura é diferente do usado neste trabalho. Assim, nós obtivemos as pontuações para uma única *thread* dos processadores usados pelo nosso trabalho e pela literatura no site https://www.cpubenchmark.net/ e fizemos uma razão entre elas, que resultou em 1,26. Assim, todos os tempos apresentados pela literatura foram divididos por este valor. Outra observação importante é que a literatura usou um tempo limite de 7200s, que divido por 1,26 resulta em aproximadamente 5714,29. Assim, este valor foi usado nos nossos experimentos como limite de tempo.

As Tabelas 1, 2 e 3 resumem os resultados comparativos para instâncias com: $|V| = 100$ e $|M| = 0$ (Tabela 1), $|V| = 100$ e $|M| > 0$ (Tabela 2) e $|V| = 200$ (Tabela 3). A coluna *Nome da Instância* apresenta o nome da instância no formato já descrito anteriormente. As colunas *Tempo(s) Este Trabalho* e *Tempo(s) Literatura* apresentam respectivamente os tempos computacionais em segundos demandados pelos algoritmos propostos neste trabalho e por Glize et al. (2020), respectivamente. A última linha de cada tabela apresenta os tempos médios considerando todas as instâncias das colunas.

Tabela 1 – Comparação com a literatura para as instâncias com $|V| = 100$ e $|M| = 0$.

	Tempo(s)			Tempo(s)	
Nome da Instância	Este Trabalho	Literatura	Nome da Instância	Este Trabalho	Literatura
A1-0-24-75-4	1,33	0,08	C1-0-24-75-4	1,05	0,16
A1-0-24-75-5	1,09	0,16	C1-0-24-75-5	1,12	0,24
A1-0-24-75-6	1,43	0,48	C1-0-24-75-6	1,40	0,40
A1-0-24-75-8	1,66	0,48	C1-0-24-75-8	4,24	1,27
A1-0-49-50-4	1,06	0,48	C1-0-49-50-4	1,13	0,71
A1-0-49-50-5	1,13	0,48	C1-0-49-50-5	1,13	0,56
A1-0-49-50-6	3,69	2,46	C1-0-49-50-6	5,27	2,46
A1-0-49-50-8	482,88	19,68	C1-0-49-50-8	7,80	17,22
B1-0-24-75-4	1,01	0,16	D1-0-24-75-4	1,02	0,16
B1-0-24-75-5	1,10	0,24	D1-0-24-75-5	1,08	0,48
B1-0-24-75-6	1,20	0,40	D1-0-24-75-6	1,10	0,24
B1-0-24-75-8	2,11	1,90	D1-0-24-75-8	1,86	1,03
B1-0-49-50-4	1,11	0,48	D1-0-49-50-4	1,04	0,56
B1-0-49-50-5	3,62	0,79	D1-0-49-50-5	1,55	0,71
B1-0-49-50-6	35,51	2,70	D1-0-49-50-6	12,24	3,17
B1-0-49-50-8	54,17	8,02	D1-0-49-50-8	36,07	96,19
Média	37,13	2,44	Média	4,94	7,85

. Fonte: Autoria própria.

Tabela 2 - Comparação com a literatura para as instâncias com $|V| = 100$ e $|M| > 0$.

	Tempo(s)			Tempo(s)	
Nome da Instância	Este Trabalho	Literatura	Nome da Instância	Este Trabalho	Literatura
A1-4-20-75-4	1,03	0,24	C1-4-20-75-4	1,02	0,16
A1-4-20-75-5	1,01	0,16	C1-4-20-75-5	1,10	0,32
A1-4-20-75-6	1,02	0,24	C1-4-20-75-6	1,30	4,84
A1-4-20-75-8	1,06	0,40	C1-4-20-75-8	1,02	0,71

A1-9-40-50-4	1,40	8,81	C1-9-40-50-4	1,12	0,71
A1-9-40-50-5	1,25	1,59	C1-9-40-50-5	1,47	13,65
A1-9-40-50-6	1,54	9,76	C1-9-40-50-6	1,49	8,89
A1-9-40-50-8	2864,71	>5714,29	C1-9-40-50-8	1,15	2,14
B1-4-20-75-4	0,98	0,16	D1-4-20-75-4	1,06	0,32
B1-4-20-75-5	1,07	0,24	D1-4-20-75-5	1,34	10,40
B1-4-20-75-6	1,11	0,40	D1-4-20-75-6	1,22	0,48
B1-4-20-75-8	1,03	0,87	D1-4-20-75-8	1,09	0,95
B1-9-40-50-4	1,08	0,48	D1-9-40-50-4	1,10	0,63
B1-9-40-50-5	1,17	1,83	D1-9-40-50-5	1,70	46,19
B1-9-40-50-6	1,13	1,35	D1-9-40-50-6	1,51	2,54
B1-9-40-50-8	1,16	1,51	D1-9-40-50-8	6,18	5,48
Média	180,11	>358,89	Média	1,55	6,15

Fonte: Autoria própria.

Tabela 3 - Comparação com a literatura para as instâncias com $|V| = 200$.

	Tempo(s)			Tempo(s)	
Nome da Instância	Este Trabalho	Literatura	Nome da Instância	Este Trabalho	Literatura
A2-0-49-150-4	1,26	1,43	B2-0-49-150-4	1,47	1,19
A2-0-49-150-5	1,87	2,22	B2-0-49-150-5	9,52	2,22
A2-0-49-150-6	12,54	2,86	B2-0-49-150-6	40,97	2,78
A2-0-49-150-8	22,57	18,97	B2-0-49-150-8	81,20	6,90
A2-0-99-100-4	5,32	16,51	B2-0-99-100-4	5,47	249,52
A2-0-99-100-5	5,28	11,03	B2-0-99-100-5	4,10	9,29
A2-0-99-100-6	759,61	23,41	B2-0-99-100-6	286,07	85,48
A2-0-99-100-8	>5714,29	656,19	B2-0-99-100-8	>5714,29	>5714,29
A2-9-39-150-4	1,20	1,43	B2-9-39-150-4	1,59	7,70
A2-9-39-150-5	1,30	3,57	B2-9-39-150-5	1,39	3,57
A2-9-39-150-6	1,57	32,38	B2-9-39-150-6	1,49	3,49
A2-9-39-150-8	1,48	7,78	B2-9-39-150-8	1,94	10,48
A2-19-80-100-4	1,68	8,10	B2-19-80-100-4	1,77	7,30
A2-19-80-100-5	12,48	3618,97	B2-19-80-100-5	28,78	3937,94
A2-19-80-100-6	13,77	>5714,29	B2-19-80-100-6	37,18	>5714,29
A2-19-80-100-8	1090,58	>5714,29	B2-19-80-100-8	2720,69	>5714,29
Média	>477,93	>989,59	Média	>558,62	>1341,92

Fonte: Autoria própria.

Observando as Tabelas 1, 2, 3 percebemos que o nosso método não resolveu até a otimalidade dentro do limite de tempo apenas 2 das 96 instâncias testadas, sendo ambas pertencentes ao grupo de instâncias maiores ($|V| = 200$) enquanto a literatura não resolveu 6 das 96 instâncias.

Outro ponto importante é que o método proposto se mostrou na média mais rápido para as instâncias das Tabelas 2 e 3, que tendem ser a mais difíceis devido as suas configurações. Uma última observação é que o nosso método teve uma maior dificuldade para maiores valores de p, em especial quando $p = 8$. Tal fato já era esperado pois, quanto maior o valor de p, maiores são as rotas viáveis para o subproblema de *pricing*, dificultando a resolução do mesmo.

A Tabela 4 apresenta as soluções ótimas de instâncias que estão sendo resolvidas até a otimalidade pela primeira vez neste trabalho.

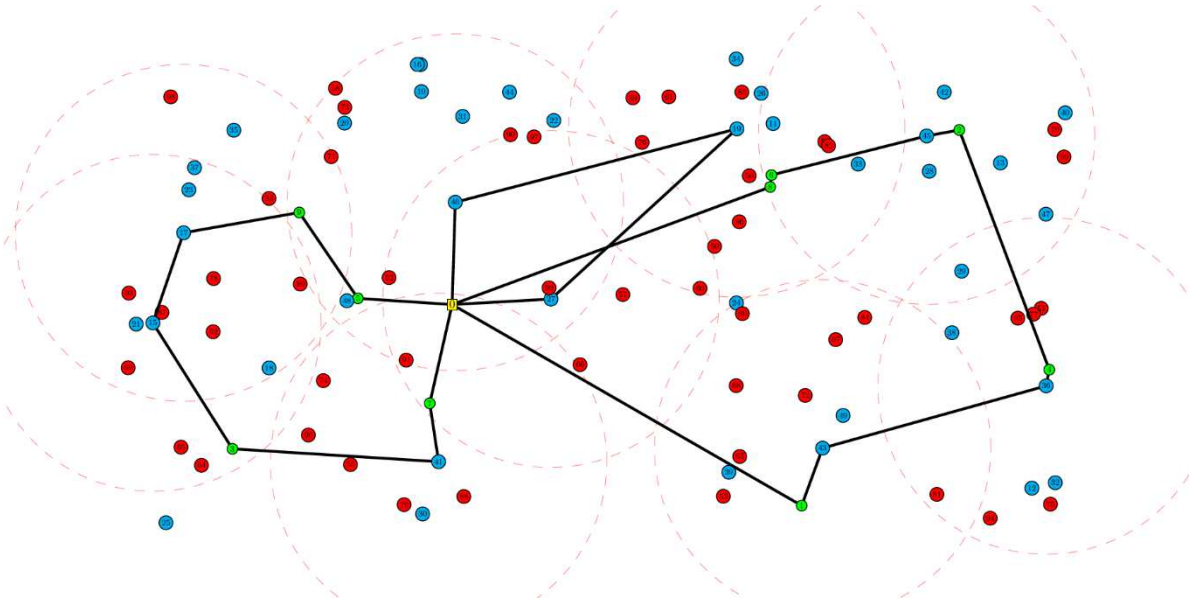
Tabela 4 – Soluções ótimas de instâncias resolvidas até a otimalidade pela primeira vez

Nome da Instância	Solução Ótima
A1-9-40-50-8	13369,0
A2-19-80-100-6	20966,0
A2-19-80-100-8	18415,0
B2-19-80-100-6	25960,0
B2-19-80-100-8	22082,0

Fonte: Autoria própria.

A Figura 1 ilustra graficamente a solução da instância A1-9-40-50-8. Nesta figura, o depósito está destacado em amarelo, os vértices mandatórios destacados em verde, os vértices opcionais destacados em azul e os vértices clientes destacados em vermelho. Além disso, as arestas que fazem parte das rotas ótimas estão destacadas em preto e, para cada ponto opcional visitado, o círculo de cobertura deste ponto foi desenhado destacados através de tracejados vermelhos opacos.

Figura 1 – Ilustração para a solução ótima da instância A1-9-40-50-8



Fonte: Autoria própria.

Observando a Figura 1, notamos que a solução ótima tem três rotas, onde apenas uma delas visitou 8 pontos, que é o máximo permitido na instância associada. Dentre os 40 pontos opcionais disponíveis, apenas 9 precisaram ser visitados.

5. Conclusões

Neste trabalho apresentamos um novo método exato para o PCMV gerado a partir da ferramenta VRPSolver. Para comprovar a robustez do método proposto, comparamos este com o melhor algoritmo exato existente para o problema. Os resultados mostraram que o método se mostrou mais rápido para boa parte das instâncias, em especial as maiores. Além disso, o método proposto comprovou a solução ótima de 5 instâncias que estavam em aberto na literatura.

Como trabalhos futuros, pretendemos estender o modelo proposto para variantes do PCMV, como por exemplo, a variante com múltiplos depósitos proposta por Allahyari et al. (2015), a variante que visa minimizar a soma dos tempos de chegada nos vértices visitados proposta por Flores-Garza et al. (2017) e a variante em que vértices clientes podem ter de ser cobertos mais de uma vez proposta por Pham et al. (2017).

REFERÊNCIAS

ALLAHYARI, Somayeh; SALARI, Majid and VIGO, Daniele. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. **European Journal of Operational Research**, v. 242, n. 3, p. 756-768, 2015.

DAMIÃO, Caio Marinho; SILVA, João Marcos Pereira; UCHOA, Eduardo. A branch-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. **4OR**, p. 1-25, 2021.

DANTZIG, George B. and RAMSER, John H. The truck dispatching problem. **Management science**, v. 6, n. 1, p. 80-91, 1959.

FLORES-GARZA, D. A.; SALAZAR-AGUILAR, M. A.; NGUEVEU, S. U. and LAPORTE, G. The multi-vehicle cumulative covering tour problem. **Annals of Operations Research**, v. 258, n. 2, p. 761-780, 2017.

GLIZE, E.; ROBERTI, R.; JOZEFOWIEZ, N. and NGUEVEU, S. U. Exact methods for mono-objective and Bi-objective multi-vehicle covering tour problems. **European Journal of Operational Research**, v. 283, n. 3, p. 812-824, 2020.

HA, M. H.; BOSTEL, N.; LANGEVIN, A. and ROUSSEAU, L. M. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. **European Journal of Operational Research**, v. 226, n. 2, p. 211-220, 2013.

HACHICHA, M.; HODGSON, M. J.; LAPORTE, G. and SEMET, F. Heuristics for the multi-vehicle covering tour problem. **Computers & Operations Research**, v. 27, n. 1, p. 29-42, 2000.

JOZEFOWIEZ, Nicolas. A branch-and-price algorithm for the multivehicle covering tour problem. **Networks**, v. 64, n. 3, p. 160-168, 2014.

KAMMOUN, M.; DERBEL, H.; RATLI, M. and JARBOUI, B. A variable neighborhood search for solving the multi-vehicle covering tour problem. **Electronic Notes in Discrete Mathematics**, v. 47, p. 285-292, 2015.

KARAOĞLAN, İsmail; ERDOĞAN, Güneş and KOÇ, Çağrı. The multi-vehicle probabilistic covering tour problem. **European Journal of Operational Research**, v. 271, n. 1, p. 278-287, 2018.

LOPES, Ramon Pereira. **Algoritmos Exatos E Heurísticos Para Problemas Seletivos De Roteamento De Veículos Com Restrições De Cobertura.** Belo Horizonte: UFMG, 2012. 62 p. Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, Universidade Federal de Minas Gerais, Belo Horizonte, 2012.

LOPES, Ramon; SOUZA, Vitor AA and DA CUNHA, Alexandre Salles. A branch-and-price algorithm for the multi-vehicle covering tour problem. **Electronic Notes in Discrete Mathematics**, v. 44, p. 61-66, 2013.

MARGOLIS, Joshua T.; SONG, Yongjia and MASON, Scott J. A multi-vehicle covering tour problem with speed optimization. **Networks**, 2021.

NAJI-AZIMI, Z.; RENAUD, J.; RUIZ, A. and SALARI, M. A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. **European Journal of Operational Research**, v. 222, n. 3, p. 596-605, 2012.

PESSOA, A.; SADYKOV, R.; UCHOA, E. and VANDERBECK, F. A generic exact solver for vehicle routing and related problems. **Mathematical Programming**, v. 183, n. 1, p. 483-523, 2020.

PHAM, Tuan Anh; HÀ, Minh Hoàng and NGUYEN, Xuan Hoai. Solving the multi-vehicle multi-covering tour problem. **Computers & Operations Research**, v. 88, p. 258-278, 2017.

QUEIROGA, Eduardo.; FROTA, Yuri.; SADYKOV, Ruslan.; SUBRAMANIAN, Anand.; UCHOA, Eduardo., and VIDAL, Thibaut. (2020). On the exact solution of vehicle routing problems with backhauls. **European Journal of Operational Research**, v. 287, n. 1, p. 76-89, 2020.