LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

# Algoritmos de Branch-Cut-and-Price para Roteamento de Veículos em Clusters

**Matheus Freitas, João Marcos P. Silva, Eduardo Uchoa**
Pós-Graduação em Engenharia de Produção - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Campus Praia Vermelha, Bloco E - sala 440, São Domingos, Niterói, RJ, Brasil, CEP: 24.210-240
matheus_freitas@id.uff.br, joaomarcossilva@id.uff.br, uchoa@producao.uff.br

**RESUMO**

O problema de roteamento de veículos em *clusters* é uma generalização do clássico roteamento com capacidade. O conjunto de clientes é particionado em *clusters*, todos os clientes no mesmo *cluster* deve ser visitados em sequencia em uma única rota. Este artigo propõe dois modelos para o problema. O primeiro deles é semelhante ao usado no clássico problema de roteamento de veículos com capacidade. O segundo tira mais proveito das características particulares do problema, pré-calculando todos os caminhos hamiltonianos mais curtos intra-*cluster*. Ambos os modelos são implementados e resolvidos pelo algoritmo de *branch-cut-and-price* existente no pacote VRPSolver. Experimentos computacionais indicam que o segundo modelo é superior. Comparando-se esse modelo com o melhor algoritmo existente na literatura, um *branch-and-cut*, observa-se resultados melhores na maioria das instâncias com mais de 100 clientes.

**PALAVRAS CHAVE. Roteamento de veículos, Geração de Colunas, Modelagem.**

**Tópicos (L&T – Logística e Transportes, PM – Programação Matemática, OC – Otimização Combinatória)**

**ABSTRACT**

The clustered vehicle routing problem is a generalization of the classic capacitated vehicle routing. The set of clients is partitioned into clusters, all clients in the same cluster must be visited in sequence by a single route. This article proposes two models for the problem. The first one is similar to the one used in the classic capacitated vehicle routing problem. The second takes more advantage of the particular characteristics of the problem, pre-calculating all the shortest intra-cluster Hamiltonian paths. Both models are implemented and solved by the branch-cut-and-price algorithm in the VRPSolver package. Computational experiments indicate that the second model is superior. Comparing this model with the best algorithm in the literature, a *branch-and-cut*, better results are observed in most instances with more than 100 customers.

**KEYWORDS. Vehicle Routing. Column Generation. Modeling.**

**Paper topics (L&T – Logistics and Transportation, PM – Mathematical Programming, OC – Combinatorial Optimization)**

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

## 1. Introdução

Este artigo aborda o problema de roteamento de veículos em *clusters* (CluVRP, *Clustered Vehicle Routing Problem*) introduzido por [Sevaux e Sörensen, 2008] como uma generalização do clássico problema de roteamento de veículos capacitado (CVRP, *Capacitated Vehicle Routing Problem*). O CVRP é definido sobre um grafo completo não direcionado $G = (V = \{0, 1, \ldots, n\}, E)$, onde o vértice 0 representa um depósito e os demais $n$ vértices os clientes. Cada cliente $i \in V \setminus \{0\}$ possui uma demanda positiva $d_i$ (por convenção, $d_0 = 0$). O depósito possui uma frota de $m$ veículos idênticos com capacidade $Q$. Cada aresta $e \in E$ possui um custo associado $c_e$. O objetivo do CVRP é determinar um conjunto de $m$ rotas com custo total mínimo, começando e terminando no depósito, atendendo a cada cliente exatamente uma vez, sem violar a capacidade do veículo. No CluVRP, é dada uma partição $\mathcal{C} = (C_1, \ldots, C_N)$ dos clientes em $N$ conjuntos disjuntos chamados de *clusters*. Existe a restrição adicional de que se um veículo visitar um cliente no *cluster* $C \in \mathcal{C}$, ele deve visitar na sequencia todos os demais clientes de $C$, para só então poder visitar outros *clusters* ou voltar para o depósito.

A motivação do CluVRP vem de dois tipos de situações. Na primeira existem restrições físicas que obrigam que todos os clientes de um *cluster* sejam atendidos em sequencia. Por exemplo, um *cluster* pode corresponder aos clientes que moram dentro de um condomínio fechado. Na segunda situação, usa-se os *clusters* para forçar que as rotas tenham alguma característica desejável, como ser mais compactas ou mais fáceis de serem aprendidas pelos motoristas.

## 2. Literatura

[Battarra et al., 2014] desenvolveram algoritmos exatos para o CluVRP e forneceram resultados para um conjunto de instâncias de *benchmark*. O algoritmo com melhor desempenho depende de um esquema de pré-processamento, que calcula o menor caminho Hamiltoniano para cada par de vértices em cada *cluster*. Os pares de vértices são as extremidades de cada caminho que atravessam o *cluster*. Isso permite selecionar um par de vértices em cada *cluster*, em vez de todo o caminho, reduzindo parte da complexidade do problema no pré-processamento. Os problemas de caminho hamiltoniano de custo mínimo resultantes são reduzidos a instâncias do problema do caixeiro viajante (TSP, Travelling Salesman Problem) e resolvidos de maneira ótima com o código Concorde [Applegate et al., 2001].

Em [Vidal et al., 2015] foram propostas três meta-heurísticas para resolução do CluVRP, duas baseadas em busca local iterada e uma delas em um algoritmo genético hibrido com representações de soluções baseadas nos *clusters*. Os três métodos produziram soluções de alta qualidade, os algoritmos que usam como vantagem a estrutura de *clusters* e vizinhanças grandes são mais rápidos. O algoritmo genético hibrido com vizinhanças grandes gera soluções de mais qualidade que os dois de busca local iterada, porém a sua fase de pré processamento pode aumentar o tempo em instancias com *clusters* grandes.

[Expósito-Izquierdo et al., 2016] propõe um método de duas fases para resolução do CluVRP, o mesmo é quebrado em dois problemas de roteamento estruturados hierarquicamente. O problema de maior nível tem como objetivo determinar a rota entre os *clusters*, enquanto o de menor nível procura a ordem de visita dos clientes em cada *cluster*. O problema de maior nível é resolvido com um algoritmo de meta-heurística e o de menor nível com técnicas exatas. Os métodos exatos no menor nível se mostraram adequados com instancias de poucos clientes, porém em instancias com muitos clientes foi necessário muito tempo computacional para resolução.De maneira geral o método apresentou todas as soluções ótimas para instancias pequenas e obteve bom desempenho em instancias com até 7500 clientes.

[Defryn e Sörensen, 2017] apresentaram uma meta heurística VNS rápida em dois níveis para resolver o CluVRP sem assumir distâncias euclideanas ou converter o problema em um CVRP

usando o método *big-M*. Ao integrar a fase de busca local do nível de clientes e do nível de *clusters* foi possível obter soluções de qualidade em pouco tempo de cálculo. No teste do algoritmo com instâncias *benchmarks* de problemas de tamanho médio e pequeno presentes na literatura em muitas foi possível chegar ao resultado ótimo com pouco tempo computacional, e quando não conseguiu foi obtido um *gap* médio de 0,03% para o ótimo. Nas instâncias grandes adaptadas do *benchmark* Golden como proposto em [Battarra et al., 2014] e [Expósito-Izquierdo et al., 2016] foram achados limitados números de ótimos, porém com *gap* médio de 1% para o ótimo em pouco tempo computacional.

Em [Hintsch e Irnich, 2018] é desenvolvida uma meta-heurística do tipo LNS *(Large Neighborhood Search)* que usa múltiplos operadores de destruição e reparação junto com um procedimento local de melhoria baseado em VND *Variable-neighborhood Descent*, com uma fase de pré processamento baseada em ILS*(Iterated Local Search)* que computa todas as rotas dentro de cada *cluster* considerando cada possibilidade de entrada e saída do mesmo.

## 3. Revisão sobre modelos VRPSolver

O VRPSolver é um pacote computacional proposto em [Pessoa et al., 2020] para facilitar a criação de algoritmos exatos para problemas de roteamento de veículos. Um modelo VRPSolver é um MIP (*Mixed Integer Program*) que contém variáveis associadas a caminhos com restrição de recursos sobre um grafo direcionado definido pelo usuário. Dado que o número de caminhos pode ser exponencialmente grande, essas variáveis são geradas dinamicamente, resolvendo-se como *pricing* problemas de caminho mais curto com restrição de recursos (RCSP, *Resource Constrained Shortest Path Problem*) [Irnich e Desaulniers, 2005]. Para garantir a integralidade das variáveis, o MIP é resolvido por um algoritmo *Branch-and-Price* (BP). Além disso, se estruturas conhecidas como *packing sets* forem definidas, cortes poderão ser adicionados, de modo que o algoritmo resultante se torna um *Branch-and-Cut-and-Price* (BCP). Nessa seção são descritos alguns dos recursos disponíveis para esses modelos. Explicações detalhadas desses, e de outros recursos omitidos, são encontradas em [Pessoa et al., 2020].

### 3.1. Grafo gerador de caminhos

Seja $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ um grafo direcionado definido pelo usuário do VRPSolver, em que $\mathcal{V}$ e $\mathcal{A}$ são, respectivamente, os conjuntos de vértices e arcos. $\mathcal{V}$ possui vértices especiais $v_{source}$ e $v_{sink}$ (podendo ser o mesmo vértice) que representam o início e o fim de um caminho com restrição de recursos. Seja $R$ o conjunto de recursos definido pelo usuário. Para cada arco $a \in \mathcal{A}$ e $r \in R$ deve existir um consumo de recurso $q_{a,r}$ e um intervalo de consumo $[l_{a,r}, u_{a,r}]$. Um caminho $p = (v_{\text{source}} = v_0, a_1, v_1, \ldots, a_{n-1}, v_{n-1}, a_n, v_n = v_{\text{sink}})$ no grafo $\mathcal{G}$ é viável se $k \geq 1, v_j \neq v_{source}, v_j \neq v_{sink}, 1 \leq j \leq k-1$ e o consumo de recurso acumulado $S^p_{j,r}$ na visita $j, 0 \leq j \leq k$ não exceder $u_{a_j,r}$, onde $S^p_{0,r}$ e $S^p_{j,r} = \max\{l_{a_j,r}, S^p_{j-1,r} + q_{a_j,r}\}$. É permitido descartar recursos para atender o limite inferior $l_{a_j,r}$, porém o limite superior $u_{a_j,r}$ deve ser respeitado.

As definições acima implicitamente definem $P$, o conjunto de todos os caminhos com restrição de recursos em $\mathcal{G}$. Para cada $a \in \mathcal{A}$ e $p \in P$, $h^p_a$ indica quantas vezes o arco $a$ aparece no caminho $p$.

### 3.2. Mapeamento e formulação

O MIP é definido pelo usuário sobre as variáveis $x_j$, $1 \leq j \leq n_1$, e $y_s$, $1 \leq s \leq n_2$. As primeiras $\bar{n}_1$ variáveis $x$ e as $\bar{n}_2$ primeiras variáveis $y$ são inteiras. As Equações (1a) e (1b) definem a função objetivo e $m$ restrições sobre essas variáveis, respectivamente. Para cada variável $x_j$, $1 \leq j \leq n_1$, o usuário deve definir $M(x_j) \subseteq A$, o seu *mapeamento* em subconjuntos não vazios de arcos. Para $a \in A$, $M^{-1}(a)$ denota o mapeamento inverso dado por $\{j | 1 \leq j \leq n_1; a \in M(x_j)\}$.

Os mapeamentos não precisam ser disjuntos, um arco pode pertencer ao mapeamento de mais de uma variável $x_j$. Como nem todo arco precisa pertencer a um mapeamento, alguns conjuntos $M^{-1}$ podem ser vazios. Pra cada caminho $p \in P$, $\lambda_p$ é definido como uma variável inteira não negativa. A relação entre essas variáveis e as variáveis $x$ é automaticamente dada por (1c). Os valores $L$ e $U$, definidos pelo usuário, são os limites inferior e superior para o número de caminhos em uma solução.

$$\text{Min} \quad \sum_{j=1}^{n_1} c_j x_j + \sum_{s=1}^{n_2} f_s y_s \tag{1a}$$

$$\text{S.t.} \quad \sum_{j=1}^{n_1} \alpha_{ij} x_j + \sum_{s=1}^{n_2} \beta_{is} y_s \geq d_i, \qquad i = 1, \ldots, m, \tag{1b}$$

$$x_j = \sum_{p \in P} \left( \sum_{a \in M(x_j)} h_a^p \right) \lambda_p, \qquad j = 1 \ldots, n_1, \tag{1c}$$

$$L \leq \sum_{p \in P} \lambda_p \leq U, \tag{1d}$$

$$\lambda_p \in \mathbb{Z}_+, \qquad p \in P, \tag{1e}$$

$$x_j \in \mathbb{Z},\ y_s \in \mathbb{Z}, \qquad j = 1, \ldots, \bar{n}_1,\ s = 1, \ldots, \bar{n}_2. \tag{1f}$$

Cada variável $x_j$, $j = 1, \ldots, n_1$, é completamente determinada por uma função linear sobre as variáveis $\lambda$ obtida através do mapeamento. De fato, único propósito dessas variáveis é permitir permitir a expressão de (1a) e (1b). Eliminando-se todas variáveis $x$ e relaxando as restrições de integralidade, obtemos o seguinte LP (*Linear Program*):

$$\text{Min} \quad \sum_{p \in P} \left( \sum_{j=1}^{n_1} c_j \sum_{a \in M(x_j)} h_a^p \right) \lambda_p + \sum_{s=1}^{n_2} f_s y_s \tag{2a}$$

$$\text{S.t.} \quad \sum_{p \in P} \left( \sum_{j=1}^{n_1} \alpha_{ij} \sum_{a \in M(x_j)} h_a^p \right) \lambda_p + \sum_{s=1}^{n_2} \beta_{is} y_s \geq d_i, \qquad i = 1, \ldots, m, \tag{2b}$$

$$L \leq \sum_{p \in P} \lambda_p \leq U, \tag{2c}$$

$$\lambda_p \geq 0, \qquad p \in P. \tag{2d}$$

O LP (2) é resolvido por geração de colunas. Seja $\pi_i$, $1 \leq i \leq m$ e $\nu_+$ and $\nu_-$, as variáveis duais das Restrições (2b) e (2c), respectivamente. O custo reduzido de um arco $a \in A$ é definido como:

$$\bar{c}_a = \sum_{j \in M^{-1}(a)} c_j - \sum_{i=1}^{m} \sum_{j \in M^{-1}(a)} \alpha_{ij} \pi_i.$$

O custo reduzido de um caminho $p = (v_0, a_1, v_1, \ldots, a_{n-1}, v_{n-1}, a_n, v_n) \in P$ é:

$$\bar{c}(p) = \sum_{j=1}^{n} \bar{c}_{a_j} - \nu_+ - \nu_-.$$

Portanto, o subproblema do *pricing* equivale a encontrar um caminho $p \in P$ com o menor custo reduzido.

### 3.3. *Packing sets* e elementos avançados

Seja $\mathcal{B} \subset 2^A$ uma coleção de subconjuntos mutualmente disjuntos de $A$ tal que as restrições:

$$\sum_{a \in B} \sum_{p \in P} h_a^p \lambda_p \leq 1, \quad B \in \mathcal{B}, \tag{3}$$

sejam satisfeitas por pelo menos uma solução ótima $(x^*, y^*, \lambda^*)$ da Formulação (1). Assim, dizemos que $\mathcal{B}$ define uma coleção de *packing sets*. A definição de $\mathcal{B}$ também é parte da modelagem e deve ser feita pelo usuário.

Os *packing sets* são usados pelo VRPSolver para melhorar a solução de (1), passando de um algoritmo de *Branch-and-Price* (BP) básico para um *Branch-Cut-and-Price* (BCP) avançado. Pessoa et al. [2020] explica como procedimentos conhecidos e aplicados para em BCP para roteamento de veículos, como *ng-paths* [Baldacci et al., 2011], *branching*, enumeração de rotas [Baldacci et al., 2008] e cortes Rank-1 [Pecin et al., 2017], são generalizados para funcionar com os *packing sets*, sendo utilizados para resolver o modelo de forma mais eficiente.

## 4. Modelos VRPSolver para o *cluster* VRP

Introduzimos a seguir algumas notações adicionais. Seja $S \subset V$, $\delta(S) \subset E$ denota o subconjunto das arestas com exatamente um extremo em $S$, enquanto $\hat{E}(S)$ é o subconjunto das arestas com ambos os extremos em $S$. Sejam $\bar{E} = \cup_{C \in \mathcal{C}} \{\delta(C)\}$ e $\hat{E} = E \setminus \bar{E}$ os conjuntos de arestas que ligam vértices de *clusters* distintos (ou um cluster e o depósito) e as arestas que ligam vértices do mesmo *cluster*, respectivamente. A demanda total de um *cluster* $C$ é dada por $d_C = \sum_{i \in C} d_i$ e o *cluster* ao qual o vértice $i$ pertence é denotado por $C(i)$. Definimos também a partição do conjunto $\mathcal{C}$ nos conjuntos $\mathcal{C}^1$ e $\mathcal{C}^2$, no qual $\mathcal{C}^1$ é o conjunto de *clusters* unitários e $\mathcal{C}^2$, os não unitários.

Nas próximas seções são apresentadas duas formulações para o problema. A Formulação $\mathcal{F}1$ é baseada na formulação do CVRP clássico [Pessoa et al., 2020], enquanto que a Formulação $\mathcal{F}2$ é baseada na suposição de que os custos dos caminhos Hamiltonianos ótimos, entre os vértices de um mesmo *cluster*, podem ser pré-computados com eficiência, como em [Battarra et al., 2014]. Por isso, denotando $\hat{c}_e, e = \{i, j\}$, como o custo do caminho Hamiltoniano ótimo que visita o *cluster* $C \in \mathcal{C}$ com extremos $i, j \in C$, podemos definir um novo conjunto de custos para as arestas: $c'_e = c_e$ se $e \in \bar{E}$ e $c'_e = \hat{c}_e$ se $e \in \hat{E}$. A Figura 1 exemplifica essa transformação, onde a Figura 1(a) mostra uma solução contendo um caminho Hamiltoniano para cada *cluster* e a Figura 1(b) ilustra a aresta única representando cada desses caminhos. Por exemplo, o caminho (0–1–7–9–10–0) é representado por (0–1–7=10–0).

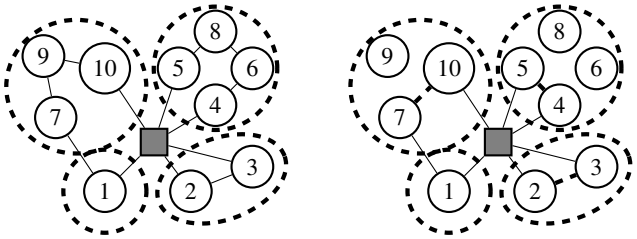


Figura 1: Representação de caminhos hamiltonianos intra-*clusters* por arestas

### 4.1. Formulação $\mathcal{F}1$

Seja o grafo $\mathcal{G} = (\mathcal{V}, \mathcal{A}), \mathcal{A} = \{(i,j),(j,i) : e = (i,j) \in E\}$ e $v_{source} = v_{sink} = 0$. O recurso de capacidade é definido como *main* ($R = R_M = \{1\}$). O consumo desse recurso em cada arco é dado por $q_a = (d_i + d_j)/2, a = (i,j) \in \mathcal{A}$ e os intervalos de consumo por $[l_i, u_i] = [0, Q], i \in V$. E temos as variáveis inteiras $x_e, e \in E$. A formulação segue:

$$(\mathcal{F}1) \quad \min \sum_{e \in E} c_e x_e \tag{4a}$$

$$\text{S.a.} \sum_{e \in \delta(\{i\})} x_e = 2, \qquad i \in \mathcal{V} \setminus 0, \tag{4b}$$

$$\sum_{e \in \delta(C)} x_e = 2, \qquad C \in \mathcal{C}, \tag{4c}$$

$$x_e \leq 1, \qquad e \in E \setminus \delta(0). \tag{4d}$$

O número de caminhos na solução é fixado no tamanho da frota de veículos ($L = U = m$); Cada variável $x_e, e = \{i,j\}$ é mapeada nos arcos $(i,j)$ e $(i,j)$, ou seja, $M(x_e) = \{(i,j),(j,i)\}$, $e = \{i,j\} \in E$. *Packing sets* são definidos nos vértices, $\mathcal{B}^{\mathcal{V}} = \cup_{i \in \mathcal{V} \setminus 0}\{\{i\}\}$. O separador de cortes de capacidade arrendondados (RCC) é dado por $(\cup_{i \in \mathcal{V} \setminus 0}\{(\{i\}, d_i)\}, Q)$ O *branching* é realizado sobre as variáveis $x$. A enumeração é utilizada.

As Restrições (4b) e (4c) garantem o grau nos vértices e *clusters*, respectivamente. A principal diferença entre $\mathcal{F}1$ e formulação clássica para o CVRP reside na inclusão das Restrições (4c), que garantem que todos os clientes do *cluster* sejam atendidos em sequência. Restrições (4d) são separadas dinamicamente. Definir o consumo de forma simétrica ($q_{(i,j)} = q_{(j,i)}$) melhora a eficiência do *pricing* [Pessoa et al., 2020]. A condição de enumeração é satisfeita sobre (4b) e (4c). Seja $S \subseteq \mathcal{B}^{\mathcal{V}}$, o separador de RCC é definido para todos os pares $(S, d(S))$ para qual $d(S) \neq 0$.

### 4.2. Formulação $\mathcal{F}2$

De início, descrevemos o grafo RCSP associado à $\mathcal{F}2$ e, posteriormente, a formulação.

#### 4.2.1. Grafo RCSP $\mathcal{F}2$

O grafo é definido por $\mathcal{G} = (\mathcal{V}, \mathcal{A}), \mathcal{V} = V, \mathcal{A} = \bar{\mathcal{A}} \cup \hat{\mathcal{A}}, \bar{\mathcal{A}} = \{(i,j),(j,i) : \{i,j\} \in \bar{E}\}$, $\hat{\mathcal{A}} = \{(i,j),(j,i) : \{i,j\} \in \hat{E}\}$ e $v_{source} = v_{sink} = 0$. Cada vértice $i \in \mathcal{V}$ possui o intervalo de consumo $[0, Q]$ e cada arco $a \in \mathcal{A}$ tem capacidade de consumo de recursos definido por (com $C(0) = 0$):

$$q_a = \begin{cases} 0, & a \in \hat{\mathcal{A}} \\ (d_{C(i)} + d_{C(j)})/2, & a = (i,j) \in \bar{\mathcal{A}} \end{cases} \tag{5}$$

A Figura 2 ilustra o grafo RCSP para o CluVRP. Cada arco duplo denota os arcos $(i,j)$ e $(j,i)$, optou-se por essa representação para tornar a figura mais clara, e aqueles tracejados representam os arcos internos de um *cluster* (ou seja, o caminho Hamiltoniano associado). Um caminho $p$ pode visitar o clientes até que o limite de capacidade $Q$ seja atingido, incluindo o consumo nos arcos na volta para o depósito. O consumo de recursos é simétrico ($q_{(i,j)} = q_{(i,j)}$), composto pelas demandas dos *clusters* $C(i), C(j)$ de cada arco $a = (i,j)$, conforme a Equação 5. Cada arco que entra e sai de $C \in \mathcal{C}$, em um caminho viável, contribui com metade da demanda $d_C$. Os arcos internos tem consumo 0.

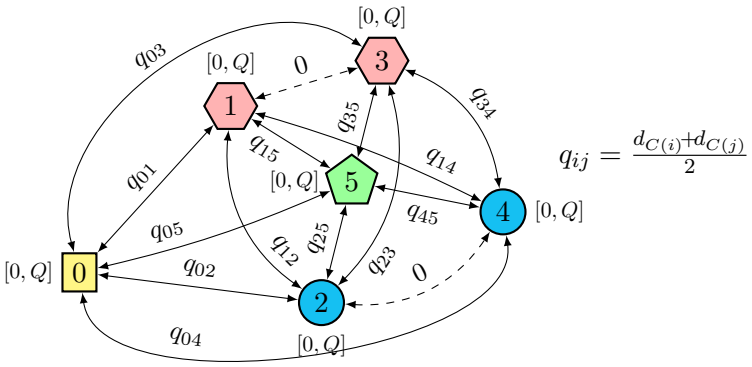


Figura 2: Grafo RCSP para o CluVRP

### 4.2.2. Formulação

Para a formulação utilizamos o grafo RCSP $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ definido na seção anterior, incluindo os mesmos consumos (Equação 5) e intervalos ($[l_i, u_i] = [0, Q]$). Os recursos de capacidade são definidos como *main* ($R = R_M = \{1\}$). E temos as variáveis inteiras $x_e, e \in \bar{E}$ e $y_e, e \in \hat{E}$. As variáveis $y$ representam o caminho Hamiltoniano dentro do *cluster*. A formulação é:

$$(\mathcal{F}2) \quad \min \sum_{e \in \bar{E}} c'_e x_e + \sum_{e \in \hat{E}} c'_e y_e \tag{6a}$$

$$\text{S.a.} \sum_{e \in \delta(C)} x_e = 2, \qquad C \in \mathcal{C}, \tag{6b}$$

$$\sum_{e \in \hat{E}(C)} y_e = 1, \qquad C \in \mathcal{C}^2, \tag{6c}$$

$$x_e \leq 1, \qquad e \in E \setminus \delta(0), \tag{6d}$$

$$x_e, y_e \in \mathbb{Z}, \qquad e \in E. \tag{6e}$$

O número de caminhos na solução é fixado no tamanho da frota de veículos ($L = U = m$); Cada variável $x_e$ ou $y_e, e = \{i, j\}$ é mapeada nos arcos $(i, j)$ e $(i, j)$, ou seja, $M(x_e) = \{(i, j), (j, i)\}$, $e = \{i, j\} \in \bar{E}$ e $M(y_e) = \{(i, j), (j, i)\}$, $e = \{i, j\} \in \hat{E}$. A coleção de *packing sets* é definida nos arcos: $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$, onde $\mathcal{B}_1 = \cup_{C \in \mathcal{C}}\{\{(i, j) : (i, j) \in \hat{E}(C)\}\}$ e $\mathcal{B}_2 = \cup_{C \in \mathcal{C}}\{\{(i, j) : (i, j) \in \delta^-(C), |C| = 1\}\}$. O *branching* é realizado sobre variáveis $x$ e $y$. A enumeração é utilizada.

A função objetivo (6a) minimiza o custo total. O conjunto de Restrições (6b) determinam que o grau de cada *cluster* seja igual a 2. As Restrições (6c) impõem que exatamente uma aresta interna (caminho Hamiltoniano) de cada *cluster*, com mais de um cliente, seja inserida na solução. Note que (6c) são suficientes para garantir o grau igual a 2 para *clusters* não unitários, e portanto, as Restrições (6b) são necessárias apenas para àqueles unitários. Porém, no caso em que instâncias utilizam, por convenção, o arredondamento do custo das arestas a desigualdade triangular pode ser violada e, portanto, é necessário incluir restrições de grau para os *clusters* não unitários. Nesse caso, essas restrições são adicionadas dinamicamente no modelo. As Restrições (6d) também são inseridas dinamicamente. A Restrições (6e) asseguram as restrições de integralidade. As Restrições sobre a quantidade de rotas (veículos) utilizadas e sua viabilidade são indiretamente garantidos pelo grafo RCSP, mapeamento das variáveis e valores de $L$ e $U$. Uma coleção $\mathcal{B}$ contém *packing sets* definidos pelos os arcos internos de cada *cluster*, e também *packing sets* definidos pelos os arcos que

entram em *clusters* unitários. A validade dos $\mathcal{B}$ escolhidos é consequência direta das Restrições (6b) e (6c) (são essenciais) e do mapeamento. Pode-se verificar que a condição suficiente de enumeração definida por [Pessoa et al., 2020] é atendida, por isso o procedimento de enumeração é usado.

A definição de $\mathcal{B}_2$ é necessária pois *clusters* unitários não estão contidos em $\mathcal{B}_1$. Caso esses não pertencessem à algum *packing set*, não poderiam ser cobertos pelas ferramentas do solver (e.g. enumeração) que atuam sobre essas estruturas e, como consequência, alguns caminhos viáveis não poderiam ser gerados. Note que, em problemas com apenas *clusters* não unitários não obteremos $\mathcal{B}_2$, e vice-versa, e ainda sim manteremos as condições necessárias para o uso dessas estruturas.

Na Formulação $\mathcal{F}1$ os *packing sets* são definidos nos vértices e, portanto, o separador de RCCs pode ser definido diretamente no modelo VRPSolver. Porém isso não é possível para *packing sets* definidos em arcos e, por isso, na Formulação 6 introduzimos, dinamicamente, os RCCs. Para tal, usamos uma rotina de separação exata através de um resolvedor MIP. Seja $\mathcal{S} \subseteq \mathcal{C}$ e $d_{\mathcal{C}} = \sum_{C \in \mathcal{C}} d_C$. Seja $\bar{x}$ a solução fracionária entre quaisquer dois *clusters*. Definimos $v_k$ como uma variável binária igual à 1 se, e somente se, o *cluster* $k$ pertence ao conjunto $S$ e $w_{kl}$ é igual à 1 se exatamente um dos *clusters* $k$ e $l$ pertencerem a $\mathcal{S}$. Para cada valor de $M = 1, \ldots, \lceil d_{\mathcal{C}}/Q \rceil - 1$ resolvemos:

$$z = \min \sum_{k,l \in \mathcal{C}} \bar{x}_{kl} w_{kl} \tag{7a}$$

$$\text{S.a. } w_{kl} \geq v_k - v_l, \qquad \forall k, l \in \mathcal{C} \cup \{0\} \tag{7b}$$

$$w_{kl} \geq v_l - v_k, \qquad \forall k, l \in \mathcal{C} \cup \{0\} \tag{7c}$$

$$\sum_{k \in \mathcal{C}} d_k v_k \geq M \cdot Q + 1, \tag{7d}$$

$$v_0 = 0, \tag{7e}$$

$$v_k \in \{0, 1\}, \qquad \forall k \in \mathcal{C}, \tag{7f}$$

$$w_{kl} \geq 0, \qquad \forall k, l \in \mathcal{C} \cup \{0\}. \tag{7g}$$

Se $z \leq 2(M+1)$, então a desigualdade de capacidade $\sum_{e \in \delta(\cup_{k \in \mathcal{S}} k)} x_e \geq \lceil \sum_{k \in \mathcal{S}} d_k/Q \rceil$ sobre o conjunto $\mathcal{S}$ é violada.

Para exemplificar a Formulação 6, considere a instância da Figura 2 com $Q = 5$, $m = 2$, $d_{C(1)} = d_{C(3)} = 3$, $d_{C(2)} = d_{C(4)} = 3$ e $d_{C(5)} = 1$. O conjunto de caminhos $P$ possui: $p_1 = ((0,1),(1,3),(3,0))$, $p_2 = ((0,1),(1,3),(3,5),(5,0))$, $p_3 = ((0,5),(5,1),(1,3),(3,0))$, $p_4 = ((0,2),(2,4),(4,0))$, $p_5 = ((0,2),(2,4),(4,5),(5,0))$, $p_6 = ((0,5),(5,2),(2,4),(4,0))$ e $p_7 = ((0,5),(5,0))$. Os custos são: $c_{01} = c_{02} = c_{12} = c_{13} = c_{34} = c_{45} = 3$; $c_{03} = c_{14} = c_{23} = 5$; $c_{04} = 6$; $c_{05} = c_{24} = 4$; $c_{15} = c_{25} = c_{35} = 2$. A formulação completa para essa instância, correspondente a Formulação 6, é:

$$\text{Min } 3x_{01} + 3x_{02} + 5x_{03} + 6x_{04} + 4x_{05} + 3x_{12} + \cdots + 2x_{35} + 3x_{45} + 3y_{13} + 4y_{24} \tag{8a}$$

$$\text{S.a. } x_{01} + x_{12} + x_{14} + x_{15} + x_{03} + x_{23} + x_{34} + x_{35} = 2 \tag{8b}$$

$$x_{02} + x_{12} + x_{23} + x_{25} + x_{04} + x_{14} + x_{34} + x_{45} = 2 \tag{8c}$$

$$x_{05} + x_{15} + x_{25} + x_{35} + x_{45} = 2 \tag{8d}$$

$$y_{13} = 1,\ y_{24} = 1 \tag{8e}$$

$$x_{01} = \lambda_1 + \lambda_2,\ x_{02} = \lambda_4 + \lambda_5,\ x_{03} = \lambda_1 + \lambda_3,\ x_{04} = \lambda_4 + \lambda_6 \tag{8f}$$

$$x_{05} = \lambda_2 + \lambda_3 + \lambda_5 + \lambda_6 + 2\lambda_7 \tag{8g}$$

$$x_{12} = 0,\ x_{14} = 0,\ x_{23} = 0,\ x_{34} = 0 \tag{8h}$$
$$x_{15} = \lambda_3,\ x_{25} = \lambda_6,\ x_{35} = \lambda_2,\ x_{45} = \lambda_5 \tag{8i}$$
$$y_{13} = \lambda_1 + \lambda_2 + \lambda_3 \tag{8j}$$
$$y_{24} = \lambda_4 + \lambda_5 + \lambda_6 \tag{8k}$$
$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 = 2 \tag{8l}$$
$$\lambda \in \mathbb{Z}_+ \tag{8m}$$

Substituindo as variáveis $x$ e $y$ e relaxando a integralidade, obtemos a Formulação 9 que pode ser resolvida por geração de colunas. Cada variável $\lambda$ representa um caminho e seu custo equivale ao custo desse caminho na solução.

$$\text{Min } 11\lambda_1 + 12\lambda_2 + 14\lambda_3 + 13\lambda_4 + 14\lambda_5 + 16\lambda_6 + 8\lambda_7 \tag{9a}$$
$$\text{S.a. } 2\lambda_1 + 2\lambda_2 + 2\lambda_3 = 2 \tag{9b}$$
$$2\lambda_4 + 2\lambda_5 + 2\lambda_6 = 2 \tag{9c}$$
$$2\lambda_2 + 2\lambda_3 + 2\lambda_5 + 2\lambda_6 + 2\lambda_7 = 2 \tag{9d}$$
$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{9e}$$
$$\lambda_4 + \lambda_5 + \lambda_6 = 1 \tag{9f}$$
$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 = 2 \tag{9g}$$
$$\lambda \geq 0 \tag{9h}$$

## 5. Experimentos

O modelo foi implementado em interface Julia 1.4 do VRPSolver [Pessoa et al., 2020]. Esse é um *framework* para a construção de algoritmos BCP para problemas de roteamento de veículos, e problemas relacionados, disponível para uso acadêmico em `vrpsolver.math.u-bordeaux.fr`. O CPLEX 12.9 é utilizado para resolver a relaxação dos LPs e os MIPs definidos sobre os caminhos enumerados. Os experimentos foram executados em um Intel® Core™ i7-3770 3.40GHz com 16 GB de RAM e sistema operacional Linux Ubuntu 16.04 LTS. Executando com apenas uma *thread* e limite de tempo de 24 horas para cada instância.

Os experimentos foram executados em conjuntos de instâncias de *benchmark* apresentados por [Battarra et al., 2014]. Os autores consideraram instâncias propostas para o problema GVRP [Bektaş et al., 2011], e também propuseram un novo conjunto adaptando as instâncias Golden do CVRP. As instâncias podem ser agrupadas de acordo com a cardinalidade do conjunto de *clusters*, dado por $|\mathcal{C}| = \lceil n/\theta \rceil$. Os conjuntos GVRP possuem 158 instâncias, variando de 16–262 clientes e de 6–100 *clusters*; o conjunto Golden é composto por 220 instâncias com 201–401 clientes e 17–97 *clusters*. As primeiras (A, B, P, M, G) são agrupadas em dois conjuntos, chamados GVRP$\theta$2 e GVRP$\theta$3 e o Golden pode ser agrupado para os valores de $\theta = \{5, \ldots, 15\}$.

### 5.1. Resultados

Os resultados são destacados em negrito de acordo com a quantidade de ótimos encontrados e o desempate é feito pelo menor tempo de execução. Fornecemos as médias aritmética e geométrica geral do tempo de execução. A média geométrica permite eliminar o efeito de *outiers* na análise dos tempos. A Tabela 1 sumariza os resultados obtidos para os grupos GVRP$\theta$2 e GVRP$\theta$3. São reportados para cada classe dos grupos GVRP2 e GVRP3, a quantidade #Ins. de instâncias, a quantidade #Opt. de ótimos obtidos e o tempo médio de execução em segundos obtidos pelas formulações $\mathcal{F}1$ e $\mathcal{F}2$. Para esse último, também são reportados os tempo de pré-processamento para obtenção dos caminhos Hamiltonianos [Battarra et al., 2014].

Tabela 1: Resultados obtidos nos grupos GVRP$\theta$2 e GVRP$\theta$3

| | | GVRP$\theta$2 | | | | | GVRP$\theta$3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\mathcal{F}1$ | | $\mathcal{F}2$ | | | $\mathcal{F}1$ | | $\mathcal{F}2$ | | |
| Conj. | #Ins. | #Opt. | Tempo méd.(s) | #Opt. | Tempo prep.(s) | Tempo méd.(s) | #Opt. | Tempo méd.(s) | #Opt. | Tempo prep.(s) | Tempo méd.(s) |
| A | 27 | 27 | 24,66 | 27 | 1,07 | **11,72** | 27 | 21,29 | 27 | 3,48 | **5,86** |
| B | 23 | 23 | **8,82** | 23 | 1,05 | 11,44 | 23 | 275,51 | 23 | 4,39 | **10,98** |
| P | 24 | 24 | 461,66 | 24 | 1,48 | **55,55** | 24 | 1173,22 | 24 | 3,17 | **10,32** |
| M-G | 5 | 4 | 374,33 | **5** | 3,00 | 4526,28 | 2 | 2053,21 | **5** | 8,00 | 115,27 |
| Total | 79 | 78 | | 79 | | | 76 | | 79 | | |
| Méd. Arit. | | | 172,38 | | | 310,69 | | 515,46 | | | 15,63 |
| Méd. Geo. | | | 4,75 | | | 7,16 | | 9,62 | | | 3,43 |

Os resultados mostram que $\mathcal{F}2$ é superior a $\mathcal{F}1$, tanto em quantidade de ótimos encontrados quanto em tempo de execução requerido (mesmo considerando o tempo necessário de pré-processamento). A formulação $\mathcal{F}1$ não foi capaz de encontrar todos os ótimos para o GVRP$\theta$3, e além disso possui tempos de processamento elevados se comparado a formulação $\mathcal{F}2$. Isso pode ser justificado pelo fato de $\mathcal{F}1$ ter desvantagem em instâncias com menor cardinalidade $|\mathcal{C}|$, caso contrário, quanto maior esse valor, suponha $|\mathcal{C}| = n$, obteríamos que as Restrições (4b) $\equiv$ (4c) e, portanto, equivaleria a formulação clássica do CVRP. Diferente da $\mathcal{F}1$, a $\mathcal{F}2$ tem vantagem sobre instâncias com menor cardinalidade, pois, as restrições de grau (6b) são definidas apenas para os *clusters*, enquanto que (6c) garantem a escolha do caminho interno desses.

Os resultados são comparados com àqueles obtidos pelo *Branch-an-Cut* (BC) proposto por [Battarra et al., 2014]. Os autores também desenvolveram um algoritmo BCP, porém este não foi capaz de resolver instâncias do conjunto Golden. O referido trabalho possui a análise do procedimento de pré-computar os caminhos Hamiltonianos, por isso a comparação é realizada desconsiderando os tempos de pré-processamento. Nos grupos GVRP$\theta$2 e GVRP$\theta$3, o BC é mais rápido que as formulações propostas com uma média de tempo de 4,90s (GVRP$\theta$2) e 5,61s (GVRP$\theta$3), porém, falha em encontrar os ótimos de 3 instâncias dos conjuntos M-G (`M-n200-k16-C100-V8`, `G-n262-k25-C88-V9`, `G-n262-k25-C131-V12`). A formulação $\mathcal{F}2$ melhorou o BKS da `G-n262-k25-C88-V9` de 3290 para 3281 e `G-n262-k25-C88-V9` de 3693 para 3683. Essa última instâncias teve um tempo de execução de 6 horas, todas as demais instância foram resolvidas em menos do que uma hora.

Considerando a limitação da $\mathcal{F}1$, reportamos apenas os resultados da $\mathcal{F}2$ as instâncias Golden. Essa execução utilizou como limite superior (UB) o BKS de [Vidal et al., 2015] acrescentado de uma unidade. Em [Battarra et al., 2014] são apresentadas algumas estratégias combinadas ao BC, reportamos os resultados do BC quando é usado o UB e um esquema de redução de grafo (GR), sendo esse o procedimento com melhores resultados. Uma vez que esses autores limitam o tempo de execução do BC em uma hora, também o fazemos. A Tabela 2 sumariza os resultados agrupados por $\theta$ quando foi dado apenas o UB. São reportados o número de ótimos e tempo médio de execução em segundos. Reportamos o tempo do BC pois a configuração do ambiente de teste é semelhante (Intel i7, 3.40 GHz).

A formulação proposta consegue resolver todas instâncias no tempo limite de 24 horas, sendo que em 8 grupos a média de tempo fica abaixo dos 7 minutos. No limite de uma hora o modelo proposto consegue provar a otimalidade em todas instâncias quando $\theta \geq 7$. No total consegue obter 3 ótimos a mais que o BC. A Tabela 3 sumariza os resultados com a redução do

Tabela 2: Resultados das instâncias Golden agregados por $\theta$ e usando apenas UB

| | $\mathcal{F}2$ | | $\mathcal{F}2$ ($<$ 1h) | | BC UB ($<$ 1h) | |
|---|---|---|---|---|---|---|
| $\theta$ | #Opt. | t méd. (s) | #Opt. | t méd. (s) | #Opt. | t méd. (s) |
| 5 | 20 | 6687,43 | 16 | 870,40 | **17** | 492,09 |
| 6 | 20 | 5851,38 | 15 | 387,08 | **17** | 185,19 |
| 7 | 20 | 407,27 | **20** | 407,27 | 17 | 202,39 |
| 8 | 20 | 157,84 | **20** | 157,84 | 19 | 184,63 |
| 9 | 20 | 106,35 | **20** | 106,35 | 19 | 140,70 |
| 10 | 20 | 78,84 | **20** | 78,84 | 19 | 84,28 |
| 11 | 20 | 59,06 | 20 | **59,06** | 20 | 242,03 |
| 12 | 20 | 46,35 | 20 | **46,35** | 20 | 83,09 |
| 13 | 20 | 38,79 | 20 | **38,79** | 20 | 64,50 |
| 14 | 20 | 34,93 | 20 | **34,93** | 20 | 101,03 |
| 15 | 20 | 30,04 | 20 | **30,04** | 20 | 68,45 |
| Total | 219 | | 211 | | 208 | |
| Méd. Arit. | | 855,90 | | 184,46 | | 162,87 |
| Méd. Geo. | | 75,98 | | 60,42 | | |

grafo. O esquema de redução visa eliminar as arestas que conectam dois *clusters* e que nunca serão utilizadas na solução ótima. Explicação detalhada é fornecida por [Battarra et al., 2014]. A coluna GR(%) contém percentual de arestas descartadas.

Tabela 3: Resultados das instâncias Golden agregados por $\theta$ e usando UB e GR

| | | $\mathcal{F}2$ GR ($<$ 1h) | | BC GR ($<$ 1h) | |
|---|---|---|---|---|---|
| $\theta$ | GR(%) | #Opt. | t méd. (s) | #Opt. | t méd. (s) |
| 5 | 72,52 | 16 | 438,46 | **17** | 363,00 |
| 6 | 80,51 | 16 | 335,13 | **19** | 86,65 |
| 7 | 85,29 | **20** | 126,16 | 19 | 109,71 |
| 8 | 88,02 | **20** | 46,47 | 19 | 93,86 |
| 9 | 89,96 | **20** | 25,03 | 19 | 92,18 |
| 10 | 90,90 | 20 | **17,06** | 20 | 48,82 |
| 11 | 91,52 | 20 | **11,16** | 20 | 72,23 |
| 12 | 91,68 | 20 | **6,92** | 20 | 48,71 |
| 13 | 91,74 | 20 | **5,23** | 20 | 40,20 |
| 14 | 91,72 | 20 | **4,39** | 20 | 39,18 |
| 15 | 91,53 | 20 | **3,65** | 20 | 23,09 |
| Total | | 212 | | 213 | |
| Méd. Arit. | | | 81,60 | | 88,66 |
| Méd. Geo. | | | 12,54 | | |

## 6. Conclusões

Este trabalho apresentou dois modelos para o CluVRP. Ambos os modelos foram implementados e testados no pacote VRPSolver. Os resultados indicam que o modelo $\mathcal{F}2$, baseado no pré-cálculo de todos os possíveis caminhos hamiltonianos dentro de cada *cluster* é superior. Comparando-se os resultados do modelo $\mathcal{F}2$ com o melhor algoritmo exato existente na literatura, o BC em [Battarra et al., 2014], observa-se o seguinte:

- Nas instâncias GVRP$\theta$2 e GVRP$\theta$3 o BC foi mais rápido para as instâncias menores, com até 100 clientes. Entretanto, para as instâncias E-M que possuem entre 100 e 262 clientes, $\mathcal{F}2$ foi melhor ao resolver mais instâncias de forma ótima.

- Nas instâncias Golden, com mais clientes porém com *clusters* bem maiores, o desempenho dos dois algoritmos foi similar em termos do número total de instâncias resolvidas e média aritmética dos tempos. Porém, observa-se que o BC foi superior para $\theta$ apenas no intervalo [5-6], enquanto $\mathcal{F}2$ foi melhor para $\theta$ no intervalo [7-15].

Finalmente, destacamos que o modelo $\mathcal{F}2$ roda diretamente no pacote VRPSolver, disponível gratuitamente para uso acadêmico. Dessa forma, o algoritmo para o CluVRP resultante pode ser utilizado por outros pesquisadores.

**Agradecimentos**

Agradecemos a Anand Subramanian por disponibilizar as instâncias e o ambiente de testes. Essa pesquisa foi parcialmente financiada pelo Comissão Nacional de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Código de financiamento 001.

**Referências**

Applegate, D., Bixby, R., Chvàtal, V., e Cook, W. (2001). Concorde TSP solver. `http://www.tsp.gatech.edu/concorde/index.html`.

Baldacci, R., Christofides, N., e Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115:351–385.

Baldacci, R., Mingozzi, A., e Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283.

Battarra, M., Erdoğan, G., e Vigo, D. (2014). Exact algorithms for the clustered vehicle routing problem. *Operations Research*, 62(1):58–71.

Bektaş, T., Erdoğan, G., e Røpke, S. (2011). Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316.

Defryn, C. e Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers & Operations Research*, 83:78–94.

Expósito-Izquierdo, C., Rossi, A., e Sevaux, M. (2016). A two-level solution approach to solve the clustered capacitated vehicle routing problem. *Computers & Industrial Engineering*, 91: 274–289.

Hintsch, T. e Irnich, S. (2018). Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, 270(1):118–131.

Irnich, S. e Desaulniers, G. (2005). Shortest path problems with resource constraints. In *Column generation*, p. 33–65. Springer.

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., e Santos, H. (2017). Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, 45(3):206 – 209.

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

Sevaux, M. e Sörensen, K. (2008). Hamiltonian paths in large clustered routing problems. In *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME*, Troyes, France.

Vidal, T., Battarra, M., Subramanian, A., e Erdogan, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research*, 58:87–99.