

Heurísticas para o problema de roteamento de veículos com atendimento suficientemente próximo

Willian Gonçalves Aparecido

Universidade Federal de Viçosa

Viçosa - MG - Brasil

willgonc@ufv.br

André Gustavo dos Santos

Universidade Federal de Viçosa

Viçosa - MG - Brasil

andre@dpi.ufv.br

RESUMO

O problema de roteamento de veículos com atendimento suficientemente próximo, também conhecido como problema do caixeiro rural direcionado generalizado com restrição de distância, é um problema de roteamento que tem como objetivo encontrar trajetos de menor custo, respeitando as restrições de tamanho da rota e, principalmente, de atendimento de consumidores. Uma aplicação da vida real interessante é a leitura de medidores de água, luz ou gás por rádio frequência através de dispositivos automáticos de leitura, onde veículos equipados com estes dispositivos precisam passar em um subconjunto de ruas para coletar as informações de todos consumidores à distância. Neste artigo, propomos uma meta-heurística híbrida com *Iterated Greedy* e *Variable Neighborhood Descent* (que utiliza quatro estruturas de vizinhanças) que, com base nas várias instâncias testadas, foi capaz de apresentar resultados parecidos com os de modelos exatos presentes na literatura, tendo como vantagem o tempo de execução utilizado.

PALAVRAS CHAVE. Problema do caixeiro rural direcionado generalizado, roteamento de veículos, meta-heurística, atendimento suficientemente próximo, rádio frequência.

Tópicos: MH - Meta-heurísticas

ABSTRACT

The close-enough cover vehicle routing problem, also known as distance-constrained generalized directed rural postman problem, is a routing problem that aims to find routes with lower cost, respecting the restrictions of route size and, mainly, of customer covering. An interesting real-life application is the reading of water, light or gas meters by radio frequency through automatic reading devices, where vehicles equipped with these devices need to pass in a subset of streets to collect information from all consumers from a distance. In this paper, we propose a hybrid metaheuristic with *Iterated Greedy* and *Variable Neighborhood Descent* (which uses four neighborhood structures) that, based on the various tested instances, was able to reach results similar to those of the exact models present in the literature, with the advantage of the execution time used.

KEYWORDS. Distance-constrained generalized directed rural postman problem, vehicle-routing, metaheuristic, close-enough covering, radio frequency.

Paper topics: MH - Metaheuristics

1. Introdução

Bem conhecido na literatura, o problema NP-difícil do carteiro rural direcionado é um problema que basicamente visa encontrar uma rota de custo mínimo que começa e termina em um posto de origem e passa por um subconjunto de ruas (pelo menos uma vez em cada rua deste subconjunto) de uma região para entrega das correspondências [Campos e Savall, 1995].

Uma aplicação interessante para este problema é a medição de leituras dos registros de luz, água e gás realizadas de porta em porta dos consumidores. Realizando uma leitura mensal em todos os consumidores, garante-se uma cobrança correta. Entretanto, nem sempre a cobrança é realizada de acordo com uma leitura apropriada. Isso pode ocorrer não somente nos casos da leitura ter sido realizada de maneira errada, mas porque o funcionário da empresa pode não ter tido acesso ao medidor, ora por não ter alguma pessoa presente no local e ora por ser uma área afastada de difícil coleta. Como o cálculo é feito por média de meses anteriores neste caso, quando uma leitura real for realizada, uma divergência grande entre as cobranças é notada. Particularmente preocupantes são os casos em que imóveis desativados passam a ser utilizados, havendo ausência de cobrança inicial durante um tempo e, após isso uma alta fatura aparece para ser quitada.

Avanços na tecnologia permitiram às companhias usarem leitores de medidas automáticos com tecnologia de identificação de rádio frequência instalados em veículos que coletam remotamente os dados enviados por dispositivos medidores, algo já utilizado nos Estados Unidos [Shuttleworth et al., 2008]. Com o uso desta tecnologia, há tanto uma garantia por parte do cliente, que sabe que sua conta não será gerada por média de consumos anteriores, como uma economia pela empresa, que poderá realocar grande parte do pessoal responsável pela leitura em outras áreas, pois este procedimento será feito mais rapidamente e de maneira mais confiável. Isto significa que as vantagens passam desde a melhoria do serviço até a redução de custo.

Além da economia a médio prazo devido à tecnologia (inicialmente haverá um gasto com a compra e instalação dos equipamentos, em contrapartida ocorre a redução do número de funcionários responsáveis por realizar as leituras), outra economia a ser realizada é em relação à ordem de visita. A tecnologia permite que cada consumidor seja atendido por visita a qualquer rua de um subconjunto dentro da sua área de cobertura, possibilitando a escolha de um trajeto menor que atenda a todos os consumidores. Assim, o uso dos dispositivos por rádio frequência economiza tempo e combustível.

O Problema de Roteamento de Veículos com Atendimento Suficientemente Próximo (PR-VASP) é uma generalização do problema do carteiro rural direcionado, no qual o conjunto de ruas que devem ser visitadas não é definido. Ao invés, um conjunto de clientes deve ser coberto por pelo menos um veículo e, como cada cliente pode ser atendido à distância (pode estar localizado em qualquer área coberta pelo conjunto de ruas, não somente na rua em si), este pode ser coberto passando por uma ou mais ruas da rota. O problema do roteamento de veículos com atendimento suficientemente próximo consiste em encontrar rota(s) de menor custo (ou tempo) que atenda(m) todos os consumidores, considerando como atendido um consumidor que esteja próximo o suficiente de alguma rota.

A primeira versão surgiu como Problema do Caixeiro Viajante Suficientemente Próximo, proposto por Shuttleworth et al. [2008], que visou resolver justamente o problema de coleta de leituras de gás, água ou eletricidade em medidores automáticos com identificação por rádio frequência. O objetivo foi criar uma rota que começasse e terminasse em um depósito e que minimizasse a distância percorrida ou tempo de viagem enquanto passasse dentro de um dado raio de distância de cada consumidor. Com 18 instâncias reais com média de 900 segmentos de ruas e 9000 consumidores cada, foram propostas quatro heurísticas para a resolução do problema. Os procedimentos têm duas fases: inicialmente um subconjunto de ruas é selecionado através de uma heurística ou programação linear inteira e, em seguida, é encontrada uma rota por meio de uma heurística sofisticada que começa e termina no depósito atravessando todas essas ruas. É reconhecido pelo autor que a solução encontrada pode ser sub-ótima, pois estas duas fases são resolvidas de modo

independente.

Posteriormente, chamando de Problema de Roteamento de Arco Suficientemente Próximo, Hà et al. [2012] apresentaram uma formulação de programação linear inteira, resolvida por um algoritmo de plano de corte. Em um trabalho posterior, Hà et al. [2014] propuseram um algoritmo *Branch-and-Cut* apresentando resultados computacionais com melhor desempenho, inclusive para instâncias de grande porte.

Com o nome de Problema Generalizado do Carteiro Rural Direcionado, Ávila et al. [2015] apresentaram duas novas formulações matemáticas, usando dois tipos de variáveis. O primeiro tipo de variável é associado com o percurso dos arcos e o segundo com os serviços necessários exigidos.

Por fim, o trabalho de Ávila et al. [2017] mudou a abordagem de resolução do problema: antes era considerado apenas um veículo e, neste trabalho, foram utilizados múltiplos veículos. Para esta versão, segundo os autores, os dois algoritmos *Branch-and-Cut* propostos foram capazes de resolver instâncias com 5 veículos, 196 vértices, 450 arcos e 150 clientes. Este problema, denominado Problema Generalizado do Carteiro Rural Direcionado com Restrição de Distância, serviu como base para este trabalho, e tem todas as características do PRVASP.

Para uma compreensão mais abrangente na área de roteamento de arcos, veja o trabalho Mourão e Pinto [2017].

Neste trabalho propomos uma heurística para o PRVASP e usamos as instâncias propostas por Ávila et al. [2017] para comprovar o bom desempenho da heurística proposta. Na seção 2 o problema é formalmente definido e uma instância exemplo é exibida. Na seção 3 mostramos a estrutura proposta para representação da solução e na seção 4 são descritos os detalhes da heurística proposta. Por fim, nas seções 5 e 6 são apresentados os resultados e a conclusão do trabalho, respectivamente.

2. Definição

Seja $G = (V, A)$ um grafo direcionado fortemente conectado, V o conjunto de vértices, cujo primeiro é considerado o depósito, e A o conjunto de arcos, representando as vias da cidade. Cada arco $(i, j) \in A$ possui um custo associado de percurso $c_{i,j} \geq 0$. É dado ainda um conjunto de consumidores W , e seja $\mathbb{H} = \{H_1, \dots, H_W\}$ uma família de subconjuntos de arcos candidatos, onde cada $H_w = \{h_w^i, h_w^{ii}, h_w^{iii}, \dots\}$, dado que $H_w \subset A$, possui os arcos responsáveis por atender o consumidor $w \in W$. Um arco candidato $h = (i, j) \in A$ pode pertencer a mais de um H , ou seja, atender a mais de um consumidor.

Com K veículos, o Problema de Roteamento de Veículos com Atendimento Suficientemente Próximo consiste em atender todos os consumidores utilizando até K rotas diferentes, que começam e terminam no depósito, sem exceder o tamanho máximo T_{max} de cada uma, de tal forma que o comprimento total dos K trajetos seja minimizado.

O bem conhecido Problema do Carteiro Rural Direcionado é um problema NP-difícil onde um subconjunto de arcos deve ser atravessado. Neste artigo trabalhamos com o PRVASP, uma generalização deste problema que, por transitividade, também é NP-difícil.

2.1. Instância

As instâncias são compostas por um grafo G representando a rede de ruas (vértices e arestas) e pelo conjunto $\mathbb{H}$ (conjuntos de arestas que podem atender cada um dos consumidores). Cada aresta possui um custo de travessia associado e pode ou não ser candidata, ou seja, atender ou não pelo menos um consumidor.

Um exemplo gráfico de instância pode ser visto na Figura 1 e, na Tabela 1a, temos a representação dos arcos candidatos responsáveis pelo atendimento de cada consumidor deste exemplo.

2.1.1. Pré-processamento

Examinando a Figura 1 e a Tabela 1a, podemos observar que quando o consumidor 3 é atendido, o consumidor 2 também é, pois todos os arcos candidatos pertencentes ao primeiro,

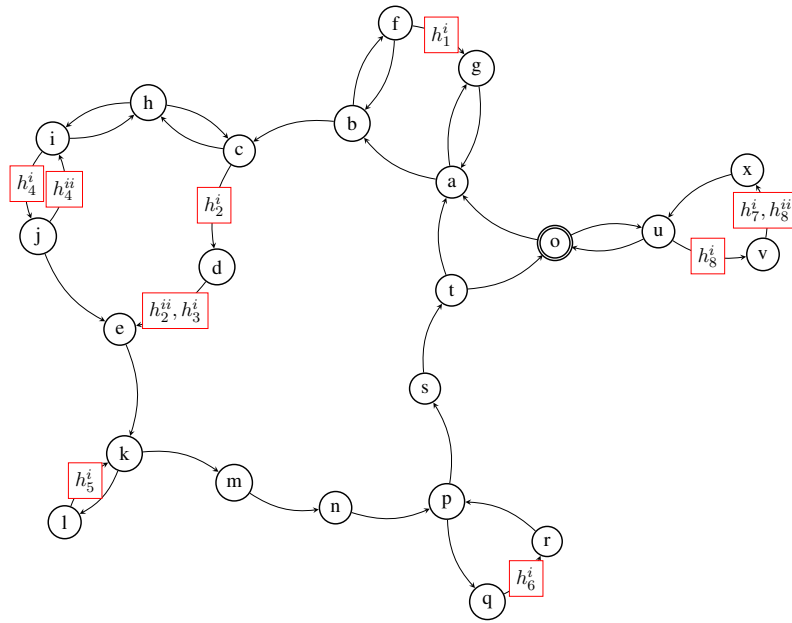
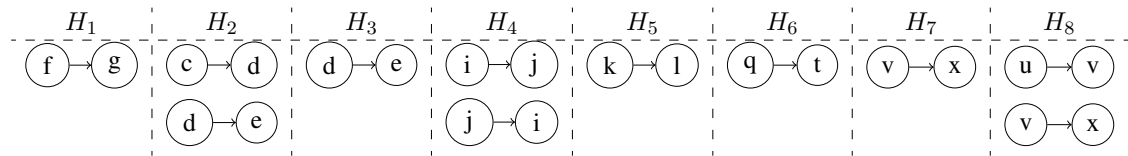
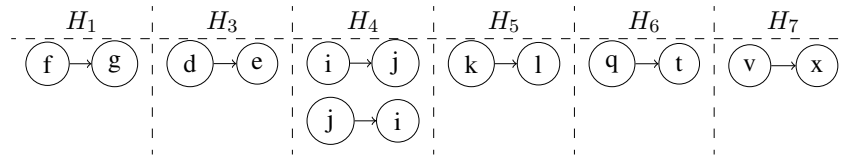


Figura 1: Exemplo de instância, os custos dos arcos não foram adicionado por motivo de clareza.



(a) Todos os consumidores e respectivos arcos candidatos



(b) Após executar o pré-processamento

Tabela 1: Conjunto de arcos candidatos por consumidor da figura 1

estão contidos no segundo. Isto ocorre com 7 e 8 também. Desta forma, podemos desconsiderar os consumidores 2 e 8, pois uma solução que atenda aos demais certamente os atende.

A partir desta análise, é feito um pré-processamento na instância logo após ter sido realizado o seu carregamento, para que assim possa-se buscar soluções de forma mais eficiente. O resultado do pré-processamento da instância de exemplo se encontra na Tabela 1b.

3. Representação e avaliação da solução

Nesta seção descrevemos a representação utilizada para uma solução e justificamos o uso da representação proposta. Descrevemos e justificamos ainda a função de avaliação adotada.

3.1. Representação

A representação de uma solução é parte essencial para a resolução do problema, pois ela afeta diretamente a forma como utilizamos as heurísticas. Esta influência está diretamente relacionada à qualidade da solução e também ao desempenho com relação ao tempo de execução.

Nos problemas de roteamento de veículos, uma solução é composta por rotas ($S = \{R_1, \dots, R_k\}$) e estas rotas normalmente são construídas a partir de uma sequência de vértices que começam e terminam na origem.

Utilizando o exemplo da figura 1, suponhamos que temos uma solução com duas rotas:

$$S = \left\{ R_1 = \{v_o, v_a, v_b, v_f, v_g, v_a, v_b, v_c, v_h, v_i, v_j, v_i, v_h, v_c, v_d, v_e, v_k, v_l, v_k, v_m, v_n, v_p, v_q, v_r, v_p, v_s, v_t, v_o\}, R_2 = \{v_o, v_u, v_v, v_x, v_u, v_o\} \mid v_i \in V \right\}$$

e um movimento *Insertion* (seção 4.2.1) faça com que o vértice v_g tenha sua posição alterada na rota R_1 para antes de v_b , assim como mostrado abaixo

$$\{v_o, v_a, v_b, v_f, \mathbf{v_g}, v_a, v_b, v_c, \dots, v_s, v_o\} \Rightarrow \{v_o, v_a, \mathbf{v_g}, v_a, v_b, v_f, v_b, v_c, \dots, v_s, v_o\}, v_i \in V$$

Este simples movimento fez com que o arco (v_f, v_g) deixasse de ser utilizado, e como consequência o consumidor 1 deixou de ser atendido.

Esta representação não é adequada para o problema pois, após qualquer perturbação realizada na rota, deve-se verificar a restrição de atendimento dos consumidores e corrigir, caso necessário.

Estes procedimentos de análise e reparação são muito custosos computacionalmente, por este motivo foi utilizada uma representação cuja restrição de atendimento dos consumidores é mais fácil de ser controlada e tais procedimentos não são necessários.

Nesta nova representação, as rotas são representadas por sequências de arcos, não de vértices, e utilizamos apenas os arcos que atendem algum consumidor (arcos candidatos). A conexão entre eles é realizada por caminho mínimo, sendo que os arcos no extremo são ligados ao vértice de origem (depósito). No mesmo exemplo utilizado, teríamos as seguintes rotas:

$$S = \left\{ R'_1 = \{h_1^i, h_4^i, h_4^{ii}, h_2^i, (h_2^{ii} = h_3^i), h_5^i, h_6^i\}, R'_2 = \{h_8^i, (h_7^i = h_8^{ii})\} \mid h_w^\delta \in H_w \right\}$$

3.2. Modificações e consequências

A representação utilizada permite que haja perturbação na ordem de visita dos arcos requeridos (arcos candidatos presentes na solução) mantendo todos os consumidores atendidos.

Quando ocorre uma perturbação, os itens da rota são religados através de caminhos mínimos e, quando um destes caminhos religados possui arcos candidatos presentes, pela nossa abordagem, estes passarão a fazer parte da nova solução. No exemplo abaixo, obtido da instância na Figura 1, a ordem de atendimento do cliente 4 da rota R'_1 foi alterada, ocasionando um aumento no número de arcos requeridos dentro da rota (os arcos h_2^i e h_2^{ii} foram visitados novamente):

$$\{h_1^i, h_4^i, h_4^{ii}, h_2^i, (h_2^{ii} = h_3^i), h_5^i, h_6^i\} \Rightarrow \{h_1^i, h_2^i, (h_2^{ii} = h_3^i), h_5^i, h_6^i, h_4^i, h_4^{ii}, h_2^i, (h_2^{ii} = h_3^i)\}$$

Esta abordagem utilizada possibilita tanto uma garantia na contagem de vezes que um consumidor foi atendido quanto no aumento de possibilidades de movimentos das heurísticas, expandindo o espaço de busca de novas soluções.

3.3. Função de avaliação

O objetivo deste problema é obter o menor custo composto de percurso de todas as K rotas respeitando as restrições de atendimento dos consumidores e o limite do tamanho de cada rota a um dado valor T_{max} .

A restrição de atendimento dos consumidores sempre é atendida, pois, conforme será visto na descrição das heurísticas, os movimentos feitos pelas heurísticas apenas movem os arcos para outro lugar na rota ou para outra rota, ou ainda trocam arcos por arcos da mesma família de subconjuntos de arcos candidatos. Porém, tais trocas e movimentos não garantem o atendimento da restrição de limite de tempo da rota. Para atender tal restrição, penaliza-se em 400% o custo da rota que exceda o limite T_{max} .

A utilização de penalidades foi escolhida; pois, além de não requerer complexos procedimentos para manter o tempo da rota dentro do limite, esta abordagem possibilita navegar em um espaço de soluções viáveis e inviáveis, aproveitando suas melhores características até encontrar a melhor solução.

4. Heurísticas

Nesta seção são detalhadas as heurísticas implementadas no presente trabalho, começando pelo algoritmo construtivo, seguido pelas estruturas de vizinhanças das buscas locais que ajudaram a construir o VND e concluindo com a heurística proposta *IG + VND*.

4.1. Heurística construtiva

O Algoritmo 1 mostra um pseudocódigo da heurística construtiva proposta, *BestArc*, explicada a seguir, que retorna uma solução satisfazendo a restrição de atendimento dos consumidores.

BestArc Semelhante ao algoritmo guloso conhecido como *Shortest available edges*, que insere a cada iteração o arco mais próximo, esta heurística analisa a adição de cada arco candidato presente em uma listagem. Entre cada par de arcos requeridos (contidos na rota) e extremos (depósito) das rotas, escolhe-se a melhor posição de inserção. A escolha da melhor posição utiliza como critério a razão entre o número de novos consumidores atendidos pela rota e o custo da rota, referentes ao novo trajeto. Após a escolha, adiciona-se apenas o arco na posição da rota escolhida e remove-se da listagem todos os arcos candidatos que não cobrem algum consumidor fora da solução, terminando o procedimento quando esta lista fica vazia. Um detalhe importante é que a adição de um novo arco preza primeiramente pela viabilidade da rota e, após isso, a escolha do melhor critério.

Algoritmo 1: Pseudocode BestArc

```

Data:  $graph = (vertices, arcs), K, T_{max}$ 
Result: Solution with all customers covered
 $routes[K] \leftarrow \emptyset;$ 
 $arcsCandidates \leftarrow getAllArcsCostumersUncovered();$ 
while  $arcsCandidates \neq \emptyset$  do
     $bestK; bestArcIn; bestPosition; bestTip \leftarrow -1; hasValidRoute \leftarrow false;$ 
    for  $k = 0$  to  $K$  do
        foreach  $arcIn \in arcsCandidates$  do
            foreach  $position \in routes[k]$  do
                 $arcP \leftarrow previousArc(position);$ 
                 $arcC \leftarrow currentArc(position);$ 
                 $cost \leftarrow calcCost(arcP.destiny \rightarrow arcIn \rightarrow arcC.origin);$ 
                 $costRemoved \leftarrow calcCost(arcP.destiny \rightarrow arcC.origin);$ 
                 $isValidRoute \leftarrow (cost + route[k].cost - costRemoved) \leq T_{max};$ 
                 $customers \leftarrow totalNewCustomersCovered(arcP.destiny \rightarrow$ 
                     $arcIn \rightarrow arcC.origin);$ 
                 $tip \leftarrow customers/cost;$ 
                if  $hasValidRoute$  and not  $isValidRoute$  then continue ;
                else if  $tip > bestTip$  then
                     $bestK \leftarrow k;$ 
                     $bestArcIn \leftarrow arcIn;$ 
                     $bestPosition \leftarrow position;$ 
                     $bestTip \leftarrow tip;$ 
                     $hasValidRoute \leftarrow isValidRoute;$ 
             $insertInRoute(routes[bestK], bestArcIn, bestPosition);$ 
         $removeUnnecessaryArcs(arcsCandidates, routes[bestK]);$ 

```

4.2. Busca Local

A partir de uma solução inicial, a cada iteração a heurística troca a solução por um vizinho que melhore a função objetivo. A busca termina quando não há candidatos melhores que a solução corrente, significando que o ótimo local foi encontrado.

O Algoritmo 2 mostra um pseudocódigo da busca local e como pode ser observado, é uma heurística determinística genérica onde basta possuir movimentos de estruturas de vizinhanças diferentes para obter soluções diferentes. Alguns desses movimentos serão apresentados a seguir, nesta subseção.

Algoritmo 2: Pseudocode local search

Data: $graph = (vertices, arcs)$, s

Result: local optima solution

while s is improved **do**

/* Return best neighbor's solution, the neighborhood varies according to the movement like ShortestPath or others */

$s_{tmp} \leftarrow doBestMovement(s)$;

if $s.getValue() > s_{tmp}.getValue()$ **then** $s \leftarrow s_{tmp}$;

4.2.1. Movimento intra-rota

Cada movimento a seguir é feito dentro de cada rota. Cada rota encontra o vizinho de menor valor (obtido através da função de avaliação) e é escolhido o vizinho que dentre todos obteve o melhor resultado.

Reduce Este movimento consiste em remover um trecho de uma rota e reconectá-la por caminho mínimo. A figura 2 mostra um exemplo, em que os arcos requeridos $\{h_9^{ii}, h_{11}^i, h_{12}^i\}$ são retirados, e os arcos candidatos $\{(h_{11}^{ii} = h_{12}^{ii}), h_{11}^{iii}\}$ são reconectados. Um movimento só é viável se não inviabiliza a solução, e de todas as rotas obtidas por tais movimentos, a de menor custo é retornada.

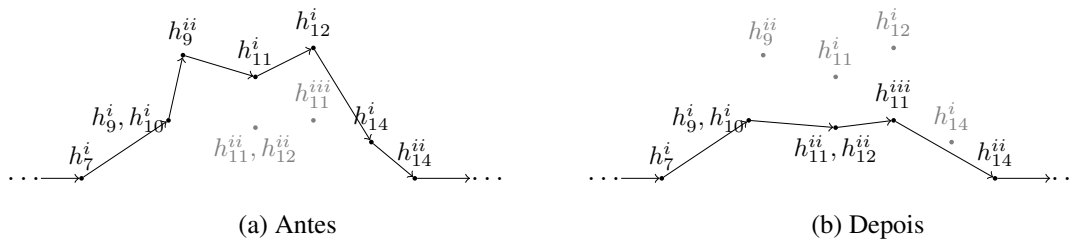


Figura 2: Troca de cobertura realizada pelo movimento *reduce*

OneDistinct A ideia principal deste movimento é remover um trecho fixo de itens ao longo da rota e fazer a melhor conexão possível com a ajuda dos arcos candidatos dos consumidores que deixarem de ser atendidos. Para identificar a melhor conexão, uma busca exaustiva é realizada por um algoritmo *backtracking* (Algoritmo 3).

Para aumentar a variabilidade, a rota resultante deve possuir pelo menos um item de diferença, ou seja, há pelo menos um arco h que não constava anteriormente na rota (daí o nome da vizinhança).

Devido ao tempo gasto com o uso da busca exaustiva, optamos pela remoção de trechos da rota definidos por 4 arcos.

Algoritmo 3: Pseudocode OneDistinct

```
Function backtraking(arcs, arcsNotAllowed, customersUncovered,  
  customer, cost) is  
  foreach arcCandidate in getArcsCustomer(customer) do  
    if arcs.size() = 0 and arcCandidate in arcsNotAllowed then continue ;  
    cost += getCost(arcs.last() → arcCandidate);  
    arcs.push_back(arcCandidate);  
    customersUncovered.remove(arcCandidate.getCustomers());  
    foreach customeraux in customersUncovered do  
      return backtraking(arcs, arcsNotAllowed, customersUncovered,  
        customeraux, cost);
```

Insertion No movimento de inserção (exemplo na figura 3), um item é removido e reinserido em outra posição da rota. Fazendo este procedimento com todos os itens da rota e em todas as posições, escolhe-se o elemento, cuja mudança de posição forneça a maior redução no custo da rota. Vale ressaltar que tal movimento pode acrescentar outros arcos candidatos à rota, aqueles que fazem parte do caminho mínimo da reconexão.

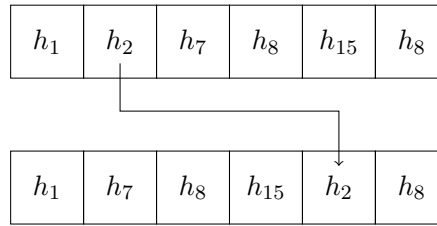


Figura 3: Reposicionamento com o movimento *insertion*

4.2.2. Movimento entre rotas

Além dos 3 movimentos intra-rota mostrados anteriormente, usamos também um movimento inter-rotas.

Exchange Como visto no exemplo da Figura 4, este movimento trabalha com pares de rotas e é realizado em duas etapas: remoção e correção. Estas duas etapas são realizadas ao longo de cada rota e entre todas as rotas contidas na solução.

Na etapa de remoção, remove-se a maior quantidade de arcos da primeira rota, de modo que haja pelo menos um arco candidato capaz de cobrir todos os consumidores descobertos, isto é, aqueles que deixaram de ser atendidos com a remoção dos arcos.

Na etapa de correção, testa-se a inserção de cada arco candidato capaz de cobrir consumidores descobertos realizando em conjunto o processo de redução de caminho visto no movimento *Reduce* em todas as posições possíveis das demais rotas.

Escolhe-se a melhor combinação de remoção e correção.

4.3. Variable Neighborhood Descent

Segundo Talbi [2009], o algoritmo determinístico *Variable Neighborhood Descent* (VND) usa sucessivas vizinhanças em busca de um ótimo local (Algoritmo 4). Primeiramente, um conjunto de estruturas de vizinhanças (que se complementam) são selecionadas e ordenadas, sendo a estratégia mais popular para a ordenação a que leva em consideração o grau de exploração e evolução de cada vizinhança. A primeira estrutura é executada enquanto há melhoria na solução, após isso, passa-se para a estrutura seguinte. Se ocorre melhoria, a primeira estrutura torna a ser executada

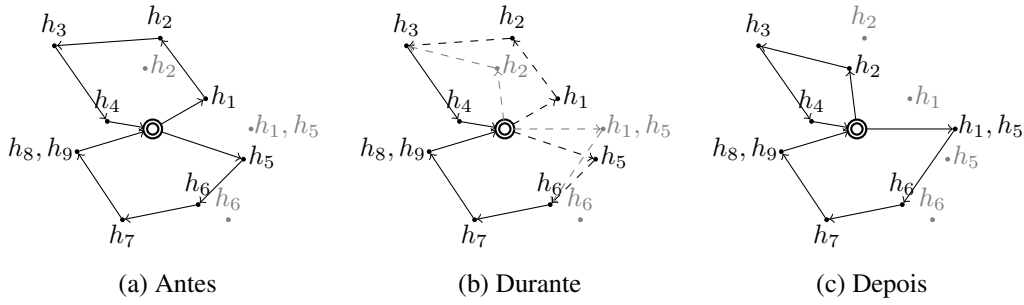


Figura 4: Movimento *exchange* entre rotas com auxílio do caminho mínimo para reconexão

novamente. O algoritmo termina quando nenhuma estrutura de vizinhança consegue melhorar a solução.

Em nossa abordagem, trabalhamos apenas com o *best improvement*, ou seja, cada vez que uma estrutura de vizinhança é explorada, ela é explorada por completo e a melhor solução vizinha é retornada.

Algoritmo 4: Pseudocode VND

Data: $graph = (vertices, arcs), s_0,$
 $localSearches[reduce, oneDistinct, insertion, exchange]$
Result: Best solution found
 $s \leftarrow s_0 ; k \leftarrow 0 ;$
while $k < localSearches.size()$ **do**
 $s_{tmp} \leftarrow localSearches[k].bestImprovement(s) ;$
 if $s.getValue() > s_{tmp}.getValue()$ **then** $s \leftarrow s_{tmp} ; k \leftarrow 0 ;$
 else $k \leftarrow k + 1 ;$

4.4. Iterated Greedy + VND

O algoritmo *Iterated Greedy* (IG) tem como característica possuir duas fases, a de destruição (retira-se uma quantidade de itens da solução de forma aleatória) e construção (que utiliza uma heurística construtiva gulosa para reparar a solução obtida da fase anterior) e a de busca local (que utiliza como ponto de partida a solução da fase anterior, de destruição/construção). A meta-heurística híbrida do IG + VND apresentada no Algoritmo 5 é simples mas poderosa, pois consegue diversificar bem com IG e intensificar com a ajuda do VND.

Algoritmo 5: Pseudocode IG + VND

Data: $graph = (vertices, arcs)$
Result: Best solution found
 $constructor \leftarrow getConstructorBestArc();$
 $s \leftarrow constructor.getSolution();$
 $s \leftarrow VND(s);$
 /* if best solution s cover customers and respect limit
 route size in less than 3 minutes, break, otherwise
 run in 6 minutes */
while $keepRunning()$ **do**
 /* During the loop, total itens removed from the
 routes varies from 20 to 50% */
 $s_{tmp} \leftarrow destruction(s);$
 $s_{tmp} \leftarrow constructor.completeSolution(s_{tmp}.getRoutes());$
 $s_{tmp} \leftarrow VND(s_{tmp});$
if $s.getValue() > s_{tmp}.getValue()$ **then** $s \leftarrow s_{tmp} ;$

Nos testes realizados, o tempo limite de 3 minutos foi suficiente para obtenção de soluções que atendessem todas as restrições para quase todas as instâncias trabalhadas. Para as poucas instâncias que falharam, foi disponibilizado um tempo extra de mais 3 minutos para que o IG + VND conseguisse encontrar soluções.

5. Resultados

Os experimentos foram realizados utilizando a linguagem de programação C++ utilizando uma *thread* por instância em um cluster que possui 2 processadores Intel(R) Xeon(R) CPU X5650 (12M Cache, 2.66 GHz, 6.40 GT/s Intel(R) QPI, 6 cores, 12 threads), 24 GB de RAM DDR3 1333 MHz, 512 GB de capacidade de armazenamento e 1 porta QDR infiniband de 40 Gbs.

Os resultados de Ávila et al. [2017] com os parâmetros K e T_{max} e 72 instâncias DCG-DRPP (que disponibilizam um total de 251 configurações diferentes) foram obtidas em <https://www.uv.es/corberan/instancias.htm>.

Utilizando como critério de parada o tempo de 180 segundos (dobrou-se o tempo caso não seja encontrada uma solução válida no limite de tempo), o método IG+VND foi executado 10 vezes para cada instância e a média dos resultados válidos obtidos por cada grupo podem ser observados nas Tabelas 2 a 5.

As tabelas possuem as seguintes colunas, na sequência: nome da instância; quantidade de veículos; resultado obtido por Ávila et al. [2017]; tempo médio das 10 execuções do IG+VND em segundos; custo médio de suas soluções; melhor custo encontrado. As duas primeiras colunas são utilizadas para identificar a configuração da instância, ou seja, com quantos veículos deve-se resolvê-la.

Tabela 2: Resultados das instâncias do repositório random50

Nome	K	Ávila	IG + VND			Nome	K	Ávila	IG + VND		
			Tempo	Custo	Melhor				Tempo	Custo	Melhor
50_3_10_1	4	7054	181	7069,0	7054	50_3_20_1	5	9211	186	9266,1	9211
	2	5477	187	5763,7	5634		2	5801	431	5953,4	5834
50_3_10_2	3	5971	181	6110,0	6110	50_3_20_2	3	6560	369	6590,2	6560
	4	6961	182	7070,8	6961		4	7484	355	7686,1	7627
50_3_10_3	3	4729	181	4843,0	4843	50_3_20_3	2	7230	183	7327,5	7230
	5	6627	181	6672,0	6627		3	7752	193	7951,7	7809
50_3_20_1	2	6958	185	6979,7	6971		4	8379	182	8553,3	8498
	3	7490	184	7528,0	7490		5	Δ 9648	188	9539,8	9418
	4	8056	182	8079,6	8056	50_3_5_1	2	4054	180	4108,8	4054

Foram omitidos os resultados para 142 instâncias em que todas as soluções foram válidas e de valor igual ou inferior a 0,01% comparado com Ávila et al. [2017], ou seja, que o resultado foi o mesmo ou praticamente o mesmo.

O símbolo delta (ex.: Δ) na terceira coluna indica que o resultado encontrado não foi declarado como ótimo em Ávila et al. [2017]. Os números sobrescritos (ex.:⁸), quando presentes, indicam quantas amostras foram utilizadas para gerar a média, ou seja, quantos resultados viáveis foram encontrados nas 10 execuções.

Todos os resultados com as rotas completas podem ser encontrados dentro da pasta *results* no site https://bitbucket.org/willgonc/dcgdrpp_viewer/src. Neste repositório também está disponibilizado um projeto de visualização gráfica para o problema feito em Javascript.

A heurística proposta (IG + VND) foi capaz de encontrar soluções viáveis para todas as 251 instâncias testadas. Das soluções apresentadas nas tabelas, 19 são melhores que as conhecidas na literatura.

É observado que apenas em 12 instâncias, cuja maioria se encontra nas Tabelas 2 e 3, a heurística utilizou em média, mais de 190 segundos, indicando o uso do tempo extra para encontrar

Tabela 3: Resultados das instâncias do repositório random75

Nome	K	Ávila	IG + VND			Nome	K	Ávila	IG + VND		
			Tempo	Custo	Melhor				Tempo	Custo	Melhor
75_3_10_1	2	6307	182	6358,3	6307	75_3_20_2	4	Δ 10684	185	10930,2	10542
	3	6807	182	6871,8	6807		5	Δ 11728	183	11606,4	11522
	5	8039	181	8214,0	8214		2	6856	202	7498,0	6929
75_3_10_2	2	8333	182	8395,9	8333	75_3_20_3	3	7082	183	7488,9	7349
	4	9323	⁹ 220	⁹ 9456,67	9323		4	7540	183	7786,0	7635
	5	9898	180	9943,0	9898		5	-	181	8560,6	8418
75_3_10_3	2	7702	194	8243,3	7995	75_3_2_1	4	6001	180	6091,0	6091
	3	8379	183	8680,0	8379	75_3_2_3	4	5276	180	5282,0	5279
	4	8933	185	9077,8	8933		5	6200	180	6406,0	6406
	5	Δ 9858	183	9755,5	9711	75_3_5_1	2	5521	180	5550,2	5521
75_3_20_1	2	8670	⁸ 212	⁸ 9990,8	9471		3	5971	180	5978,2	5971
	3	9081	⁵ 372	⁵ 9865,6	9422	75_3_5_2	3	5371	181	5389,8	5371
	4	Δ 9952	² 187	² 10159,5	9858		4	6891	181	6969	6969
	5	Δ 10615	296	11032,6	10481		5	8028	181	8331	8290
75_3_20_2	2	9461	183	9816,0	9589	75_3_5_3	2	4226	180	4281,2	4226
	3	9833	205	10577,1	10010						

Tabela 4: Resultados das instâncias do repositório Albaida

Nome	K	Ávila	IG + VND			Nome	K	Ávila	IG + VND		
			Tempo	Custo	Melhor				Tempo	Custo	Melhor
A3107_55	3	4222	180	4294,0	4222	A3205_107	5	10526	180	10587,0	10526
	4	4803	180	4818,0	4803	A3207_45	2	3719	180	3723,0	3719
A3111_41	3	21132	180	21282,9	21132	A3209_92	2	7896	180	7897,5	7896
	2	6994	180	6998,1	6994	A5109_37	5	9967	³ 240	³ 9967,0	9967
A3203_7	4	9845	180	9993,2	9845	A5201_41	3	5505	180	5509,5	5505

soluções viáveis. Dessas 12 instâncias, em 5 não foi encontrado solução em todas as execuções. Não foram encontradas, entretanto, características que as diferenciavam das demais.

6. Conclusão

Este trabalho abordou o problema de roteamento de veículos com atendimento suficientemente próximo, onde foi apresentada uma representação de solução mais adequada e, em cima desta representação foi proposta uma meta-heurística híbrida do *Iterated Greedy* com *Variable Neighborhood Descent*, sendo o VND composto pelas buscas locais *Reduce*, *OneDistinct*, *Insertion* e *Exchange*.

As instâncias e os melhores resultados obtidos por Ávila et al. [2017], encontrados por algoritmos exatos *branch-and-cut*, foram utilizados na comparação com o IG + VND. A heurística proposta conseguiu resultados equiparáveis aos da literatura na grande maioria dos casos, encontrou soluções viáveis para todos (pior caso foram 2 em 10 tentativas) encontrou soluções viáveis para instâncias em que não se conhecia nenhuma. Para algumas instâncias presentes no trabalho foi disponibilizado o tempo de execução e entre elas algumas apresentaram tempo inferior ao tempo limite do IG + VND, fato ocorrido também com a meta-heurística porém o objetivo principal foi encontrar soluções em tempo hábil para todas as instâncias.

Dessa forma, acredita-se que utilizando o IG para explorar e o VND para intensificar a busca por soluções pode servir como boa abordagem para o problema de roteamento de veículos com atendimento suficientemente próximo. Como trabalho futuro, pretende-se utilizar uma nova meta-heurística em conjunto com VND adicionando mais alguma estrutura de vizinhança de forma a reduzir ainda mais o tempo de execução, garantido a qualidade da solução.

Agradecimentos

Os autores agradecem às instituições CAPES e FAPEMIG por proporcionarem os recursos necessários para a realização deste trabalho. Pesquisa desenvolvida com o auxílio do Cluster da

Tabela 5: Resultados das instâncias do repositório Madrigueras

Nome	K	Ávila	IG + VND			Nome	K	Ávila	IG + VND		
			Tempo	Custo	Melhor				Tempo	Custo	Melhor
M3101_129	2	7499	180	7773,8	7499	M3203_140	4	Δ 10388	180	10050,6	10000
	3	8115	180	8244,5	8115		5	-	180	10456,1	10445
	4	-	180	8921,4	8919	M3205_55	2	9180	180	9268,3	9180
	5	-	180	10275,7	10272		3	9742	180	9859,7	9742
M3103_05	3	Δ 9052	180	9253,1	8994		4	Δ 11526	180	10128,8	10097
	4	Δ 9628	180	9564,6	9539		5	-	180	10679,8	10638
	5	-	180	9774,0	9774	M3207_123	2	5464	180	5544,5	5464
M3105_187	2	7852	180	8586,0	7852		3	5762	180	5846,4	5762
	3	Δ 9454	180	9130,1	9086		4	6160	180	6162,3	6160
	4	-	180	9108,3	9086	M3209_81	2	10069	180	10091,6	10069
	5	-	180	9404,6	9320		3	10577	180	10618,4	10577
M3107_93	5	Δ 5681	180	5115,0	5115		4	11305	180	11346,3	11305
M3109_156	2	8983	180	9120,7	8993		5	Δ 11933	180	11960,9	11920
	5	Δ 10794	180	10862,1	10808	M3211_178	2	27798	180	27993,9	27798
M3111_139	2	25328	180	25404,0	25328		3	31967	180	32396,6	31967
	3	26552	180	26581,6	26552		4	Δ 38869	180	37639,4	37421
	4	Δ 30997	180	29449,3	29422		5	-	180	38302,0	38150
	5	Δ 29686	180	29431,1	29422	M5103_03	4	9284	180	9286,6	9284
M3201_157	2	9032	180	9143,6	9032	M5107_121	2	4012	180	4030,6	4012
	3	Δ 10317	180	9844,2	9667	M5207_08	3	4769	180	4770,0	4769
	4	Δ 10451	180	10373,6	10152	M5209_142	2	6069	180	6471,3	6069
	5	Δ 12224	180	10885,3	10554		3	6900	180	6949,8	6900
M3203_140	2	8997	180	9101,7	8997	M5211_23	4	Δ 8834	180	8888,5	8834
	3	9595	180	9892,8	9595		3	18099	180	18121,5	18099

UFV (Universidade Federal de Viçosa).

Referências

- Ávila, T., Corberán, Á., Plana, I., e Sanchis, J. M. (2015). A new branch-and-cut algorithm for the generalized directed rural postman problem. *Transportation Science*, 50(2):750–761.
- Ávila, T., Corberán, Á., Plana, I., e Sanchis, J. M. (2017). Formulations and exact algorithms for the distance-constrained generalized directed rural postman problem. *EURO Journal on Computational Optimization*, 5(3):339–365.
- Campos, V. e Savall, J. (1995). A computational study of several heuristics for the drpp. *Computational optimization and applications*, 4(1):67–77.
- Hà, M. H., Bostel, N., Langevin, A., e Rousseau, L.-M. (2012). An exact algorithm for the close enough traveling salesman problem with arc covering constraints. In *1st International Conference on Operations Research and Enterprise Systems (ICORES)*, p. 233–238.
- Hà, M. H., Bostel, N., Langevin, A., e Rousseau, L.-M. (2014). Solving the close-enough arc routing problem. *Networks*, 63(1):107–118.
- Mourão, M. C. e Pinto, L. S. (2017). An updated annotated bibliography on arc routing problems. *Networks*, 70(3):144–194.
- Shuttleworth, R., Golden, B. L., Smith, S., e Wasil, E. (2008). Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, p. 487–501. Springer.
- Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.