An Application of ILS heuristic to Periodic Vehicle Routing Problem with Heterogeneous Fleet and Fixed Costs

Robert Cristian Abreu
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, MG, Brasil
Email: robert.abreu@ufv.br

José Elias C. Arroyo
Departamento de Informática
Universidade Federal de Viçosa
Viçosa, MG, Brasil
Email: jarroyo@dpi.ufv.br

***Abstract*—This paper addresses the Periodic Vehicle Routing Problem (PVRP) which is a variation of the classical Vehicle Routing Problem. The PVRP consists of assigning a combination of visiting days to each customer, and defining the set of routes for each day of a planning horizon. The objective of the problem is the minimization of the total length of the routes travelled by the vehicles and the fixed costs of using the vehicles on the time horizon. To solve the problem, we present an Integer Programming Model and two heuristic algorithms based on the meta-heuristic Iterated Local Search (ILS). The mathematical model is solved by using the CPLEX optimization solver and optimal solutions are obtained for small and medium-sized instances of the problem. The computational experiments show that the heuristic algorithms determine high quality approximate solutions at a low computational time. The results are validated by statistical tests.**

Keywords - Periodic Vehicle Routing, Combinatorial Optimization, Integer Programming, Meta-heuristics

I. Introdução

O Problema de Roteamento de Veículos (PRV) é um dos clássicos problemas abordados na área de Otimização Combinatória. Estudado pela primeira vez em 1959 [1], o problema consiste em determinar rotas de custo mínimo para um determinado conjunto de veículos homogêneo de modo que eles atendam um conjunto de clientes, respeitando as restrições impostas. Este tipo de problema tem ganhado espaço e desde então diversas variações foram criadas a fim de se adequar as diferentes características dos modelos de transporte do mundo real, como por exemplo veículos capacitados, janelas de tempo, múltiplos depósitos, dentre outros ([2], [3]).

Uma das classes do PRV, chamada de Problema de Roteamento de Veículos Periódico (PRVP), considera um determinado período de planejamento, geralmente de alguns dias, para que sejam realizados os atendimentos aos clientes[4]. Neste período, chamado de horizonte de planejamento, os clientes precisam ser visitados de acordo com suas necessidades. Dessa forma, cada um deles determina em quais dias eles precisam ser atendidos, criando assim suas respectivas agendas de visita.

O PRVP pode ser utilizado no planejamento de diversas atividades reais como a coleta de lixo ([5], [6], [7]), recolhimento de leite por laticínios ([8], [9]), entrega de roupas em hospitais [10], distribuição de produtos para supermercados ([11], [12]) e até mesmo distribuição de sangue [13]. Esses problemas são em sua grande maioria de difícil solução visto que, sendo um problema da classe NP-Difícil [14] não existem algoritmos exatos que forneçam uma solução ótima em tempo computacional viável.

Por se tratar de um problema da classe NP-Difícil, métodos heurísticos são as técnicas propostas para sua resolução. Dentre os métodos já utilizados destacam-se as heurísticas Busca Tabu de Cordeau [15], VNS de Hemmelmayr [16], uma heurística híbrida baseada em cobertura de conjuntos proposta Cacchiani [17] e o GRASP de Pacheco [12]. Porém, todos estes autores abordam o problema padrão do roteamento periódico sem levar em consideração os custos fixos envolvidos no processo de transporte.

Os custos fixos são os custos associados à utilização e manutenção dos veículos disponíveis para o atendimento dos clientes [18]. O custo fixo esta geralmente associado à capacidade total do veículo. Menor capacidade implica em menor custo fixo. Isso decorre, pois a capacidade está diretamente relacionada ao custo de aquisição e depreciação [19]. O custo fixo é pago caso o veículo seja utilizado no atendimento aos clientes, em qualquer dia do horizonte de planejamento.

Para aproximar o modelo do problema de roteamento periódico às situações reais, este trabalho considera que a frota disponível para atender os clientes é formada por veículos com características diferentes, chamada frota heterogênea. Nesse problema, além das decisões sobre roteamento, é necessário que se escolham os veículos mais adequados de modo que se obtenha o menor custo envolvido. Cada veículo possui uma capacidade máxima de carga e um custo fixo a ser pago pelo seu uso.

Neste trabalho, para resolver o PRVP, são propostas duas heurísticas baseadas na meta-heurística *Iterated Local Search* (ILS). A primeira é uma heurística ILS convencional e a segunda é um ILS que utiliza o método *Random*

978-1-4673-9143-6/15/$31.00

Variable Neighborhood Descent (RVND) como busca local. Também é proposto um modelo de Programação Inteira que é resolvido utilizando o software de otimização CPLEX.

O artigo é organizado da seguinte maneira. Na Seção II são apresentados a descrição do problema e a modelagem matemática. As heurísticas propostas são apresentadas na Seção III. A Seção IV apresenta os resultados dos testes computacionais realizados e finalmente as conclusões do trabalho estão na Seção V.

II. Descrição do problema

O PRVP é uma generalização do clássico PRV onde o horizonte de planejamento é de t dias. Para cada dia é necessário criar o roteamento dos clientes que precisam ser visitados neste dia. Seja $U = \{u_0, u_1, ..., u_n\}$ o conjunto de locais onde u_0 corresponde ao depósito e os demais n locais correspondem aos clientes. Define-se um grafo completo $G = (U, E)$, sendo U o conjunto de vértices e $E = \{(i, j) : i, j \in U, i \neq j\}$ o conjunto de arestas. Cada cliente i possui um conjunto de agendas de visitas R_i e uma demanda q_i, idêntica para todos os dias de sua agenda. Seja $V = \{v_0, v_1, ..., v_w\}$ o conjunto de veículos disponíveis no depósito, cada veículo possui um custo fixo f_v a ser pago caso o veículo seja usado.

O problema consiste em determinar as rotas dos veículos para cada dia do horizonte de planejamento de tal forma que as demandas dos clientes sejam atendidas e os custos operacionais de transporte sejam minimizadas. As seguintes condições devem ser satisfeitas: Toda rota começa e termina no depósito. Um veiculo deve ser ser utilizado uma vez por dia e sua capacidade deve ser respeitada. O tempo máximo de duração de cada rota deve ser respeitado. Todos os clientes devem ser atendidos de acordo com sua agenda de visita ou dias de disponibilidade. Cada cliente só pode ser atendido por um único veículo por dia.

Com base na formulação apresentada por [5], neste trabalho é apresentado um modelo de Programação Linear Inteira (PLI) para o PRVP com frota de veículos heterogêneos e com custos fixos. O modelo utiliza os seguintes índices, parâmetros e variáveis de decisão.

Índices:

- $i; j$: Clientes
- v: Veículos
- l: Dias

Parâmetros:

- n: Número total de clientes.
- w: Número total de veículos disponíveis.
- t: Número total de dias do horizonte de planejamento
- d_{ij}: Distância euclidiana entre os clientes i e j.
- q_i: Demanda do cliente i.
- f_v: Custo fixo do veículo v.
- Q_v: Capacidade do veículo v.
- T_v: Distância máxima para a rota de um veículo v.
- L: Conjunto de dias do horizonte.
- V: Conjunto de veículos disponíveis.
- U: Conjunto de locais $U = \{u_0, u_1, ..., u_n\}$ onde u_0 corresponde ao depósito.
- U_c: Conjunto de clientes $U_c = \{u_1, u_2, ..., u_n\}$.
- R_i: Conjunto de agendas possíveis para o cliente i.
- $a_{rli} = \begin{cases} 1, & \text{se o dia } l \text{ pentence à agenda } r \text{ do cliente } i \\ 0, & \text{caso contrário.} \end{cases}$

Variavéis de decisão:

- $x_{ijvl} = \begin{cases} 1, & \text{se o veiculo } v \text{ no dia } l \text{ viaja de } i \text{ para } j \\ 0, & \text{caso contrário.} \end{cases}$
- $y_{ir} = \begin{cases} 1, & \text{se a agenda } r \text{ foi escolhida para o cliente } i \\ 0, & \text{caso contrário.} \end{cases}$
- $Y_v = \begin{cases} 1, & \text{se o veiculo } v \text{ é utilizado} \\ 0, & \text{caso contrário.} \end{cases}$

O modelo de PLI é apresentado a seguir:

$$\text{Minimizar} \sum_{v \in V} \sum_{l \in L} \sum_{i \in U} \sum_{j \in U} d_{ij} x_{ijvl} + \sum_{v \in V} f_v Y_v$$

Sujetio a:

$$(1) \sum_{r \in R_i} y_{ir} = 1 \quad \forall i \in U_c$$

$$(2) \sum_{j \in U, j \neq i} \sum_{v \in V} x_{ijvl} - \sum_{r \in R_i} a_{rli} y_{ir} = 0 \quad \forall i \in U_c; \forall l \in L$$

$$(3) \sum_{i \in U, i \neq h} x_{ihvl} - \sum_{j \in U, j \neq h} x_{hjvl} = 0 \quad \forall h \in U_c; \forall l \in L; \forall v \in V$$

$$(4) u_{ivl} - u_{jvl} + Q_v x_{ijvl} \leq Q_v - q_j \quad \forall i, j \in U_c; j \neq i; \forall l \in L; \forall v \in V$$

$$(5) q_i \leq u_{ivl} \leq Q_v \quad \forall i \in U_c; \forall l \in L; \forall v \in V$$

$$(6) \sum_{j \in U_c} x_{0jvl} \leq 1 \quad \forall l \in L; \forall v \in V$$

$$(7) \sum_{i \in U} \sum_{j \in U} x_{ijvl} \geq Y_v M \quad \forall l \in L$$

$$(8) \sum_{i \in U_c} x_{i0vl} = \sum_{j \in U_c} x_{0jvl} \quad \forall l \in L; \forall v \in V$$

$$(9) \sum_{i \in U} \sum_{j \in U, j \neq i} (t_{ij} + s_i) x_{ijvl} \leq T_v \quad \forall l \in L; \forall v \in V$$

$$(10) x_{ijvl}, y_{ir}, Y_v \in \{0, 1\}; \quad i, j \in U; l \in L; v \in V; r \in R_i$$

O objetivo é minimizar a distância percorrida e os custos fixos associados à utilização dos veículos. As restrições (1) asseguram que uma agenda adequada é atribuída para cada cliente. As restrições (2) garantem que os clientes são visitados apenas nos dias correspondentes em sua agenda. As restrições (3) garantem que toda vez que um veículo chega a um determinado cliente, em seguida ele deixa tal cliente. As restrições (4) e (5) garantem que não há repetições na rota. As restrições (6) garantem que cada veículo seja utilizado no máximo uma vez por dia. As restrições (7) determinam quais veículos são usados para atender os clientes, sendo M uma constante suficientemente grande. As restrições (8) garantem que todo veículo que sai do depósito deve retornar ao mesmo. As restrições (9) garantem que o tempo máximo de cada rota será respeitado. E por fim as restrições (10) determinam o domínio das variáveis.

III. Heurística Iterated Local Search

Iterated Local Search (ILS) [20], é uma meta-heurísticas bastante utilizada na literatura para solucionar problemas de otimização NP-Difícil. A ideia desta meta-heurística é utilizar um método de busca local que explora o espaço de soluções por meio de perturbações de soluções ótimas locais geradas durante a busca. De acordo com Tang [21], o ILS se destaca por ser uma meta-heurística eficiente e simples.

Na implementação de um algoritmo ILS, quatro passos devem ser especificados. Primeiramente, o processo de construção da solução inicial `CriaSolucao()`, em seguida o procedimento de busca local, `BuscaLocal()`, que recebe a solução atual e retorna um ótimo local. O terceiro, o procedimento de perturbação, `PerturbaSolução()`, que altera alguma característica da solução corrente para que com isso o processo de busca local consiga explorar um espaço de busca maior. E por fim o critério de aceitação de novas soluções.

Além dos quatro passos citados anteriormente, este trabalho adiciona mais dois procedimentos ao método ILS. A primeira se trata do reinicio da solução atual. Após um determinado número de iterações, se a solução atual não apresentar melhora no valor da função objetivo, ela será reiniciada. Ou seja, a solução atual será descartada e uma nova solução será gerada pelo método construtivo `CriaSolucao()`. O segunda procedimento é um método de busca local chamado `OtimizaAgendas()`. Este método será chamado sempre que a solução atual não apresentar melhora após a execução do procedimento `BuscaLocal()`. O pseudocódigo da heurística ILS implementado neste trabalho é apresentado no Algoritmo 1.

No Algoritmo 1 tem-se que a solução inicial s é gerada na linha 2 e já atribuída a melhor solução conhecida $s*$. A solução atual s é melhorada na linha 5 pelo procedimento de busca local, retornando assim a solução s' que é o ótimo local obtido a partir de s. Nas linhas 6 a 10 o critério de aceitação é avaliado. A solução atual s e a melhor solução

Algorithm 1 ILS()

```
    iterSemMelhora = iterAtual = 0;
 2: s = CriaSolucao();
    s* = s;
 4: while (iterAtual < maxIter) do
        s' = BuscaLocal(s);
 6:     if (Custo(s') < Custo(s)) then
            s = s';
 8:         if (Custo(s) < Custo(s*)) then
                s* = s;
10:         end if
        else
12:         iterSemMelhora = iterSemMelhora + 1;
            OtimizaAgendas(s);
14:     end if
        if (iterSemMelhora == maxIterSemMelhora) then
16:         s = Construtivo();
            iterSemMelhora = 0;
18:     else
            s = PerturbaSolucao(s);
20:     end if
        iterAtual = iterAtual + 1;
22: end while
    Retorne s*;
```

encontrada pelo algoritmo $s*$ são atualizadas caso o valor da função objetivo de s' seja menor. Caso contrário, na linha 13, o processo de otimização de agendas de visita é chamado. Nas linhas 15 a 17 tem-se o reinicio da solução atual. Na linha 19 a solução atual é perturbada. Por fim na linha 23 a melhor solução encontrada após todas a iterações do método é retornada. Nas próximas seções os elementos principais do ILS estão detalhados.

A. Construção da solução inicial

O método construtivo utilizado é um método totalmente aleatório. Para todo cliente i do conjunto de clientes é atribuída uma agenda de visitas aleatória pertencente ao seu conjunto de agendas disponíveis Ri. Em seguida, o cliente i é inserido nos dias correspondentes na agenda de visita escolhida em um veículo aleatório pertencente ao conjunto de veículos V. O pseudocódigo do método construtivo pode ser visto no Algoritmo 2.

Algorithm 2 CriaSolucao()

```
    Inicializa a solução s com todas as rotas dos veículos vazias.
 2: for (todo cliente i em U_c) do
        Selecione uma agenda de visita r aleatória pertencente a R_i
 4:     for (todo dia l em L) do
            if (cliente i precisa ser visitado no dia l na agenda r) then
 6:             Selecione um veículo v aleatório pertencente ao conjunto de
                veículos V de s
                Insira i na rota de v do dia l na última posição.
 8:         end if
        end for
10: end for
    Retorne s;
```

Esse método pode gerar soluções inviáveis, visto que, como a escolha do veículo é aleatória, o veículo pode receber clientes mesmo que sua capacidade máxima ou distância

máxima sejam ultrapassadas. Por esse motivo a utilização de penalidades foi adotada. A utilização dessas penalidades ao avaliar as soluções fará com que durante o processo de otimização das rotas elas também sejam guiadas para a situação de rotas válidas.

B. Avaliaçao de soluções

O método construtivo utilizado, bem como as perturbações de soluções, podem fazer com que uma solução se torne inviável, ou seja, a restrição de capacidade máxima dos veículos ou a distância máxima de percurso em uma rota seja violada. Para evitar que o algoritmo ILS retorne uma solução inviável, é utilizado um método para penalizar as soluções inviáveis. Este método foi proposto por Cordeau [22]. Uma penalidade α é aplicada sobre o excesso de carga e uma penalidade β é aplicada sobre o excesso de distância. Dessa forma o cálculo da função objetivo de uma solução pode ser definido como:

Função objetivo (Custo de uma soluçao):
$\sum_{v \in V} \sum_{l \in L} \sum_{i \in U} \sum_{j \in U} d_{ij} x_{ijvl} \ + \sum_{v \in V} f_v Y_v + \alpha \times Q_{extra} + \beta \times D_{extra}$

Onde, Q_{extra} é o excesso de carga:
$Q_{extra} = \sum_{v \in V} max[(\sum_{i \in U_c} \sum_{j \in U_c} x_{ijvl} q_j) \ - Q_v; 0]$
$l \in L$

D_{extra} é o excesso de distância:
$D_{extra} = \sum_{v \in V} max[(\sum_{i \in U_c} \sum_{j \in U_c} x_{ijvl} d_{ij}) \ - T_v; 0]$
$l \in L$

A utilização destas penalidades conduz o algoritmo a soluções viáveis uma vez que o valor da função objetivo de uma solução inviável tenderá a ser maior do que o valor das soluções viáveis.

C. Buscas locais

O procedimento `BuscaLocal()` é baseada em por movimentos (vizinhanças) clássicos do PRV. Para cada dia do horizonte de planejamento esses movimentos são aplicados a todas as rotas existentes no dia. Foram utilizadas quatro estruturas de vizinhanças inter-rota e três estruturas intra-rota, definidos abaixo:

Inter-Rota

- Shift(h, h): A partir da rota 1, h clientes, são realocados para a rota 2, mantendo a ordem de visita original.
- Swap(h, h): Duas rotas terão uma sequência de h clientes trocados entre si.
- Cross: Um arco (i, j) pertencente a rota 1 e outro arco (i', j') pertencente a rota 2 são removidos e dois novos arcos são inseridos, sendo eles (i', j) e (i, j').
- Radial: Para todo cliente i da rota 1 é feita uma verificação para determinar se existe algum outro cliente j de outra rota que está dentro de um determinado raio centrado em i. Caso exista, este cliente j será inserido na rota 1 na posição adjacente a i de menor custo.

Intra-Rota

- Swap(1): Dois clientes da mesma rota têm suas posições trocadas entre si.
- 2-Opt: Dois arcos da rota são removidos e outros dois são inseridos de modo que haja uma inversão na ordem de visita dos clientes envolvidos nestes arcos.
- Or-Opt(1): Um cliente da rota é removido e reinserido na posição mais vantajosa.

Todos os movimentos só serão efetuados caso nenhuma restrição seja violada e caso a mudança leve a uma melhora na função objetivo.

Com os movimentos definidos acima, para o procedimento `BuscaLocal()` foram implementados dois algoritmos diferentes. O primeiro algoritmo de busca local, chamado `BLEstática()` utiliza uma ordem fixa de aplicação dos movimentos. Esta ordem foi definida por testes empíricos considerando primeiro os movimentos iter-rota (com apenas com $h = 1$) e em seguida os movimentios intra-rotas. Os melhores resultados foram obtidos com a seguinte sequência de movimentos: Inter-rota Shift(1,1) -> Swap(1,1) -> Radial -> Cross e em seguida as intra-rota Swap(1) -> Or-Opt(1) -> 2-Opt.

O segundo algoritmo de busca local é baseada na heurística VND (*Variable Neighborhood Descent*) com sua característica aleatória, ou seja, a ordem de aplicação das vizinhanças é determinada aleatoriamente a cada iteração. Esta variação do VND é chamada de RVND (*Randomized VND*). Neste trabalho, ao iniciar o algoritmo `RVND()`, uma estrutura de vizinhança aleatória é escolhida dentre as possíveis no conjunto de movimentos Iter-Rota. Caso haja melhora na solução após a aplicação dos movimentos da vizinhança escolhida, a mesma estrutura será aplicada novamente na solução modificada, caso não haja melhora, a estrutura escolhida será removida do conjunto e uma nova será selecionada aleatoriamente dentre as restantes. O algoritmo `RVND()` termina quando o conjunto de buscas inter-rota fica vazio.

O conjunto de vizinhanças utilizado para o `RVND()` contém seis elementos. Utilizamos Shift(u, u) e Swap(u, u) com duas versões cada, uma considerando u igual a 1 e outra com u igual a 2. As duas restantes são as demais buscas inter-rota, Cross e Radial. A vizinhança gerada por cada um desses elementos consiste na utilização do movimento da busca local aplicado exaustivamente na solução até que nenhuma melhora seja encontrada. Se após a utilização de

alguma dessas vizinhanças a solução apresentar melhora, ela será submetida ao processo de buscas locais intra-rota. Neste processo as buscas Swap(1), 2-Opt e Or-Opt serão utilizadas em todas as rotas da solução que sofreram alguma alteração. O pseudocódigo do algoritmo `RVND` é apresentado no Algoritmo 3.

Algorithm 3 RVND(Solução s)

```
    Inicialize a lista de vizinhanças LV com as seis vizinhanças inter-rota
 2: while (LV ≠ ∅) do
        Selecione aleatoriamente uma vizinhança k pertencente a LV
 4:     Encontre o melhor vizinho s' pertencente a vizinhança k(s)
        if (Custo(s') < Custo(s)) then
 6:         Aplique os movimentos intra-rota nas rotas alteradas em s'
            s = s';
 8:     else
            Remova a vizinhança k do conjunto LV
10:     end if
    end while
12: Retorne s;
```

Na linha 1 o conjunto de vizinhanças LV é inicializado com as seis vizinhanças inter-rota, Shift(1,1), Shift(2,2), Swap(1,1), Swap(2,2), Cross() e Radial(). Na linha 3 uma dessas vizinhanças é escolhida aleatoriamente para ser aplicada a solução s. Na linha 4 o melhor vizinho s' obtido após a aplicação da vizinhança escolhida na solução s é recuperado. Se s' é melhor que s, ou seja, se o valor da função objetivo de s' é menor, então os movimentos intra-rota serão aplicados em s'. A ordem de aplicação dos movimentos intra-rota é o mesmo utilizado na BLEstática. Se não houve melhora em s após a aplicação da vizinhança escolhida, esta é removida da lista LV na linha 9. O algoritmo termina quando o conjunto LV estiver vazio.

D. Perturbação

Uma fase essencial do ILS consiste na perturbação da solução atual. No caso do PRVP, uma estratégia eficiente de perturbação consiste em alterar as agendas de visita atribuídas aos clientes visando aumentar o espaço de busca explorado. Três métodos de perturbação foram implementados.

A primeira perturbação, chamada `Perturbação1`, consiste em alterar a agendas de visita de uma determinada porcentagem de clientes para a próxima agenda na lista. Considere a lista de agendas de visita possíveis como uma lista circular. Esta porcentagem é calculada sobre o número de clientes que podem ter sua agenda alterada, ou seja, aqueles que possuem mais de uma agenda de visita disponível. A porcentagem de alteração foi de 10%, sendo o número mínimo de clientes alterado igual a 1, a menos que a instância não possua nenhum cliente com mais de uma opção de agenda de visitas.

A segunda perturbação, chamada `Perturbação2`, é muito semelhante à primeira. A diferença está na prévalidação da nova agenda do cliente antes de alterá-la, ou seja, a agenda do cliente escolhido somente será alterada caso ela não torne a solução atual inviável pela capacidade. Em outras palavras, é feito o cálculo da nova demanda total dos clientes, considerando a nova agenda escolhida para o cliente atual. Se a soma da demanda de todos os clientes de um determinado dia for superior a soma das capacidades máximas de todos os veículos disponíveis, esta agenda é descartada e um novo cliente é escolhido. Graças ao fator aleatório da escolha dos clientes e a própria configuração de determinadas instâncias, este método tem um limite máximo de tentativas por iteração igual a 50% do número total de clientes da instância.

A terceira perturbação, chamada de `Perturbação3`, irá alterar um número x de clientes baseado no número k de iterações sem melhora da solução atual. Desse modo, x clientes, sendo x o mínimo entre k e o máximo entre 20% do número total de clientes ou o número total de clientes que possuem mais de uma agenda de visita, terão sua agenda atual de visita alterada. Em outras palavras, x pode ser definido como $x = min(k; max(0, 2 * n; |U_r|))$ sendo U_r o conjunto de todos os clientes em que $|R_i| > 1$.

E. Procedimento Otimiza Agendas

Após a escolha aleatória das agendas de visita no método construtivo e o processo de sucessivas perturbações, é possível que a solução fique presa em um grupo de ótimos locais, impossibilitando que um espaço de busca maior seja explorado. Dessa forma, sempre que, ao completar um determinado número de iterações, a solução não apresentar melhora após a perturbação e a aplicação das buscas locais, um processo de otimização de agendas será executado. Este processo pode ser considerado um método de busca local aplicado aos dias de visita e não apenas as rotas.

A otimização de agendas consiste em verificar para todo cliente $i \in U_c$, em ordem aleatória, se existe alguma outra agenda de visita que levaria a um ganho no custo da solução. Assim, para todas as agendas disponíveis para o cliente i é feita uma simulação de troca da agenda atual pela próxima na lista na solução corrente. Caso existam agendas melhores que a atual, aquela que obtiver o menor custo será a agenda escolhida. O pseudocodigo do método `OtimizaAgendas()` pode ser visto no Algoritmo 4.

No Algoritmo 4, na linha 1, é criada uma lista auxiliar contendo todos os clientes existentes. Nas linhas 4 e 5 o valor da função objetivo da solução atual e a agenda de visita atual do cliente i são armazenados. Nas linhas 6 a 14 é feita a simulação da alteração da agenda de visitas do cliente i e o melhor valor obtido é armazenado, juntamente com qual agenda conseguiu este valor. Na linha 16 a agenda de i será alterada em s, caso exista alguma agenda que melhore a solução atual. O processo de simulação possui basicamente duas etapas, remover o cliente das rotas nos dias aos quais ele não pertence mais e adicionar o cliente as novas rotas. A adição do cliente nas novas rotas é feita testando a inserção do novo cliente em todas as posições possíveis das rotas

já existentes no novo dia e selecionado aquela em que o aumento no custo seja o mínimo.

Considere, por exemplo, o cliente i com suas respectivas agendas de visita $R_i = \{\{1, 2\}, \{2, 3\}\}$. Isso significa que na agenda 1 o cliente i deve ser visitado nos dias 1 e 2, e na agenda 2 o cliente i deve ser visitado nos dias 2 e 3. Suponha que na solução atual, a agenda escolhida para i seja a agenda 1. Quando o cliente i for escolhido durante o método de otimização, uma simulação será feita trocando a agenda atualmente escolhida, no caso a 1, por todas as outras disponíveis. Neste caso, i possui apenas mais uma agenda, então a agenda 1 será trocada pela 2. Isso implica que o cliente i será removido da rota em que ele estiver no dia 1, não sofrerá alteração no dia 2 e será inserido na rota de menor custo no dia 3. Se esta movimentação acarretar em uma redução no custo total da solução, a agenda atual do cliente i será trocada para a 2, caso contrário a 1 será mantida. Este processo é feito para todos os clientes em ordem aleatória.

Algorithm 4 OtimizaAgendas(Solução s)

```
    Inicialize a lista C com todos os clientes existentes em U_c.
 2: while (C ≠ ∅) do
        Selecione aleatoriamente um cliente i ∈ C.
 4:     Inicialize o valor de bestFO com o Custo(s).
        Inicialize bestR com a agenda atualmente escolhida para i.
 6:     for (toda agenda r ∈ R_i) do
            if (r não for a agenda atualmente escolhida para i) then
 8:             Faça a simulação da troca da agenda atual de i para r em s
                retornando a nova solução s'.
                if (Custo(s') < bestFO) then
10:                 bestFO = Custo(s');
                    bestR = r;
12:             end if
            end if
14:    end for
       if bestFO < Custo(s') then
16:        Troque a agenda atual de i por bestR em s.
       end if
18:    remova o cliente i de C.
    end while
20: Retorne s;
```

IV. Testes e resultados

Nesta seção estão detalhados os testes realizados e os resultados obtidos. Os algoritmos ILS e ILS-RVND foram comparados com resultado obtido pelo CPLEX versão 12.6, na resolução do modelo matemático, com tempo máximo de execução limitado a uma hora para cada instância testada. Como para grandes instâncias o tempo necessário para que uma solução fosse encontrada pelo CPLEX é muito alto, o número de clientes foi limitado para no máximo 30. Os algoritmos ILS foram implementados em C++ e todos os testes executados em um computador i7-4790K com 32GB de memória RAM.

A. Geração de instâncias de teste

Para o problema abordado neste trabalho foram criadas no total 198 instâncias descritas a seguir. O número n de clientes varia de 10 a 30 crescendo de 2 em 2, o que totaliza onze grupos de instâncias separadas pelo número de clientes. Cada grupo possui 18 instâncias. Para cada grupo, a localização dos clientes e o depósito (pontos) foram distribuídos em um plano cartesiano de 100x100 de duas formas: na metade das instâncias os pontos foram dispostos de forma totalmente aleatória, incluindo o depósito. Na outra metade o depósito foi colocado na posição central do plano (50,50) e os clientes divididos em *clusters*, sendo cada *cluster* um quadrante do plano. O número de clientes em cada quadrante deve ser igualmente distribuído. Por exemplo, para as instâncias com 20 clientes serão escolhidos 5 para cada quadrante. Para as instâncias em que a divisão $n/4$ não seja exata o número de clientes em cada quadrante será o mais próximo possível.

As demandas q_i de cada cliente foram atribuídas aleatoriamente dentro do intervalo de 50 a 100. A capacidade Q_v dos veículos foi dividida em duas categorias, uma restrita e uma com folga. Nas instâncias com capacidades restritas a soma da capacidade dos veículos é a soma total das capacidades demandadas pelos clientes mais uma pequena sobra de 1% a 5%. Na classe de instâncias com folga nas capacidades, até dois veículos a mais foram adicionados fazendo com que a soma das capacidades dos veículos chegasse ao dobro das capacidades demandadas em alguns casos. Essa diferença visa observar o comportamento dos métodos de resolução para determinar se no caso restrito pelo menos uma solução será encontrada e, se no caso com folga os veículos mais dispendiosos serão excluídos do roteamento.

O custos fixo (f_v) de um veículo v é proporcional a sua capacidade. o valor de f_v gerado aleatoriamente no intervalo $[Q_v, 1.5 \times Q_v]$.

A distância máxima de percurso de um veículo v (T_v) é definida como a distância total de uma rota contendo todos os clientes ordenados de forma aleatória.

O horizonte de planejamento L determinado para todas as instâncias foi de 2, 4 e 6 dias. O número de dias de visita de cada cliente foi determinado aleatoriamente entre 1 e L. As agendas de visitas dos clientes foram escolhidas aleatoriamente dentro das possibilidades existentes para o número de dias escolhido e o tamanho do horizonte de planejamento. Por exemplo, para um horizonte de quatro dias, com três dias de visita, as opções de visitas seriam {1,2,3}, {2,3,4}, {1,2,4}, {1,3,4} e {2,3,4}.

B. Testes computacionais

No algoritmo ILS dois tipos de buscas locais foram implementadas. O ILS padrão utiliza a busca local chamada de BLEstática e será chamado simplesmente de ILS. O outro algoritmo utiliza como busca local a heurística RVND, este algoritmo é chamado de ILS-RVND. A comparação do desempenho dos algoritmos heurísticos será medida calculando o RPD (*Relative Percentage Deviation*) dos resultados, com

relação a melhor solução conhecida. O RPD é determinado da seguinte maneira:

$$RPD = \frac{C_{corrente} - C_{melhor}}{C_{melhor}} \times 100$$

Onde $C_{corrente}$ é o custo da solução obtida pelo algoritmo para uma determinada instância e C_{melhor} é o melhor custo conhecido obtido dentre todos os algoritmos para uma determinada instância.

Para cada algoritmo ILS foram testadas as três perturbações propostas. Os algoritmos foram executados dez vezes para cada instância. O melhor resultado e a média das dez execuções foram armazenadas. Com os valores do RPD obtidos a partir destes resultados foi executado um teste estatístico de Análise de Variância (ANOVA). O resultado deste teste para o ILS está no gráfico da figura 1 e para o ILS-RVND no gráfico da figura 2.

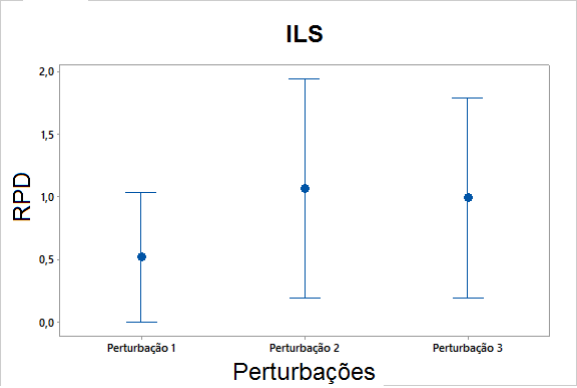

Figure 1. Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação do método de perturbação para o ILS.

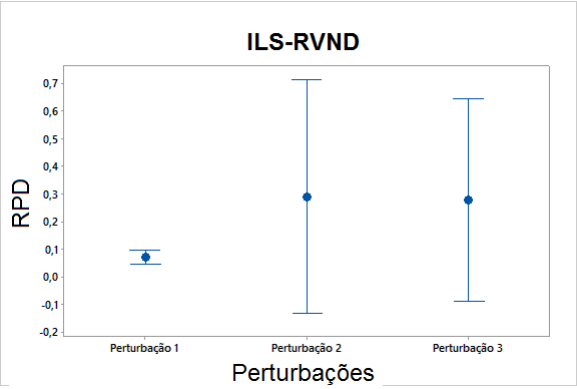

Figure 2. Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação do método de perturbação para o ILS-RVND.

Pelos gráficos é possível observar que tanto para o ILS quanto para o ILS-RVND a perturbação 1 apresenta melhores médias do RPD. No ILS, a diferença entre as perturbações não são significativas mas com a perturbação 1 obtem-se a menor média. Já no ILS-RVND, a primeira perturbação apresenta uma variação pequena em relação as outras duas, o que a qualifica como a mais estável das três.

Com os resultados obtidos nos testes anteriores a perturbação escolhida para ser utilizada nas comparações com o método exato foi a perturbação 1. A seguir, comparam-se os algoritmos ILS e ILS-RVND como o CPLEX que resolve o modelo matemático do problema. Vale ressaltar que a execução do CPLEX, para cada instância, foi limitada a uma hora. Se a execução do CPLEX finaliza pelo tempo limite (1 hora), significa que a solução retornada por este método não necessariamente é a ótima. Para o cálculo do RPD, o melhor valor da função objetivo é o menor dentre os obtidos pelos três métodos, CPLEX, ILS e ILS-RVND.

Nas Tabelas I e II são exibidas as médias dos valores do RPD para os métodos comparados em cada grupo de instâncias. Na tabela I estão os resultados das comparações das melhores soluções obtidas pelos algoritmos ILS e ILS-RVND (nas 10 execuções) com as soluções obtidas pelo CPLEX. Na tabela II estão as comparações dos resultados médios dos algoritmos. Observa-se que nas duas tabelas as médias encontradas pelos algoritmos ILS e ILS-RVND são muito inferiores as obtidas pelo CPLEX. E dentre os dois algoritmos, o ILS-RVND possui médias ligeiramente menores.

Nº clientes	CPLEX	ILS	ILS-RVND
10	3,38	0,00	0,00
12	4,30	0,21	0,08
14	0,67	2,15	0,06
16	3,60	1,72	2,23
18	7,06	0,04	0,01
20	8,32	0,05	0,02
22	25,96	0,31	0,30
24	13,37	0,06	0,13
26	34,57	0,02	0,04
28	29,99	0,04	0,05
30	23,07	0,08	0,10
Média:	14,03	0,43	0,27

Table I
Médias dos RPDs (por grupo de instâncias) dos algoritmos comparados para o melhor resultado encontrado.

Para validar os resultados obtidos pelos algoritmos e verificar se as diferenças observadas são estatisticamente significantes, foi realizada uma Análise de Variância (ANOVA) paramétrica. Ainda, foram verificadas as três pressuposições da ANOVA para que os resultados do teste sejam estatisticamente válidos: normalidade, igualdade de variância e a independência dos resíduos. Dado que o valor-P na ANOVA resultou em 0,00, e este valor é menor que 0,05, pode-se concluir que as diferenças são estatisticamente significativas.

Os resultados da ANOVA são apresentados nas figuras 3 e 4. Essas figuras mostram o gráfico de médias e intervalos

Nº clientes	CPLEX	ILS	ILS-RVND
10	3,38	0,57	0,21
12	4,30	4,73	1,07
14	0,67	4,96	0,83
16	3,60	2,84	3,03
18	7,06	2,88	1,37
20	8,32	1,63	1,32
22	25,96	5,76	5,54
24	13,37	3,29	3,10
26	34,57	1,57	1,12
28	29,99	1,68	1,00
30	23,07	3,08	1,97
Média:	14,03	3,00	1,87

Table II
Médias dos RPDs (por grupo de instâncias) dos algoritmos comparados para a média de dez execuções.

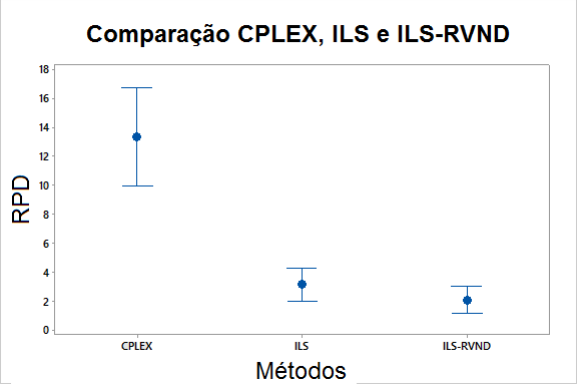

Figure 4. Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação das médias dos resultados encontrados pelos três métodos.

HSD de Tukey com nível de confiança de 95% para as comparações das melhores soluções (3) e das médias (4) dos algoritmos ILS e ILS-RVND.

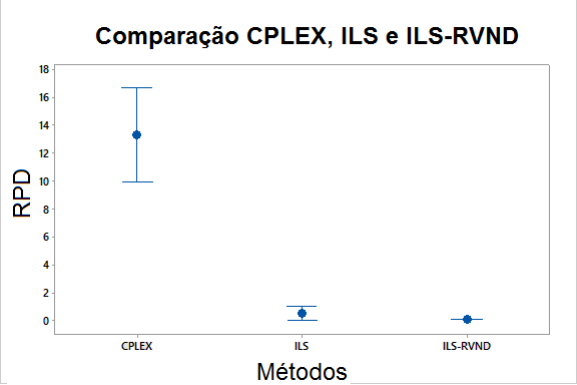

Figure 3. Gráfico de Médias e intervalos HSD de Tukey com nível de confiança de 95% para a comparação dos melhores resultados encontrados pelos três métodos.

Nessas figuras observa-se a grande diferença dos resultados encontrados pelos métodos ILS e ILS-RVND em relação ao CPLEX. Do total de 198 instâncias testadas o ILS conseguiu um resultado igual ou melhor que o CPLEX em 190 instâncias e o ILS-RVND em 192. Além disso, o tempo gasto pelos algoritmos é muito inferior ao método exato. O tempo médio gasto pelo ILS foi de 4,24 segundos e para o ILS-RVND 5,86 segundos. A execução do CPLEX finalizou antes do tempo limite de uma hora para apenas 7 instâncias. Ou seja, o CPLEX conseguiu provar a otimalidade nessas 7 instâncias. Para essas instâncias, ambos os algoritmos ILS conseguiram encontrar as soluções ótimas.

Os dados da figura 4 representam o resultado médio de dez execuções. Neste caso, o ILS conseguiu um resultado igual ou melhor ao CPLEX em 161 instâncias e o ILS-RVND em 169.

V. Conclusão

Neste trabalho foi abordado o Problema de Roteamento de Veículo Periódico com custos fixos dos veículos. Para resolver o problema foram desenvolvidos, um modelo de Programação Inteira e dois algoritmos heurísticos baseados na meta-heurística ILS. Os testes computacionais mostraram que o algoritmo ILS apresentou ser um algoritmo eficiente e estável para a resolução do problema.

As heurísticas ILS e ILS-RVND conseguiram resultados iguais ou melhores que o CPLEX em 95,95% e 96,96% do total das instâncias, respectivamente. Os valores da função objetivo encontrados pelos algoritmos heurísticos ILS e ILS-RVND, em comparação aos valores obtidos pelo CPLEX, em média foram 8,53% e 9,01% menores, respectivamente. Além disso, as heurísticas necessitaram de, no máximo, 10,25 segundos para resolver uma instância, já o CPLEX, para resolver o modelo matemático e obter uma solução viável, gastou 3600 segundos, sendo que em apenas 7 instâncias gastou um tempo nenor a ese tempo limite. Estes resultados evidenciam a qualidade dos métodos heurísticos.

Pela complexidade existente na resolução do modelo matemático do problema, as comparações puderam ser feitas apenas com instâncias pequenas, com no máximo 30 clientes e 6 dias de planejamento, porém como trabalhos futuros pretende-se estudar o comportamento dos algoritmos heurísticos na resolução de instâncias maiores do problema. Como o ILS e o ILS-RVND apresentaram resultados bastante favoráveis, acredita-se que estes algoritmos sejam eficientes para resolver instâncias maiores.

Acknowledgments

The authors would like to thank CNPq, FAPEMIG and CAPES.

References

[1] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[2] P. Toth and D. Vigo, "The vehicle routing problem, society for industrial and applied mathematics," *SIAM Monographs on Discrete Mathematics and Applications*, 2002.

[3] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, 2009.

[4] N. Christofides and J. E. Beasley, "The period routing problem," *Networks*, vol. 14, no. 2, pp. 237–256, 1984.

[5] W. Higino, H. M. Bezerra, E. J. Araújo, K. C. Poldi, and A. A. Chaves, "Metaheurísticas simulated annealing e pesquisa em vizinhança variável aplicadas ao problema de roteamento periódico de veículos para coleta de lixo," *XVI CLAIO/XLIV SBPO, Rio de Janeiro*, 2012.

[6] S. Baptista, R. C. Oliveira, and E. Zúquete, "A period vehicle routing case study," *European Journal of Operational Research*, vol. 139, no. 2, pp. 220–229, 2002.

[7] E. Angelelli and M. G. Speranza, "The application of a vehicle routing model to a waste collection problem: two case studies," in *Quantitative Approaches to Distribution Logistics and Supply Chain Management*. Springer, 2002, pp. 269–286.

[8] I. C. V. da Silva, R. P. Reis, and M. J. N. Gomes, "Custos e otimização de rotas no transporte de leite a latão e a granel: um estudo de caso," *Organizações Rurais & Agroindustriais*, vol. 2, no. 1, 2011.

[9] W. A. do Valle and T. H. Nogueira, "Análise e redução dos custos logísticos da coleta de leite a granel pela utilização de um modelo de roteamento."

[10] M. Banerjea-Brodeur, J.-F. Cordeau, G. Laporte, and A. Lasry, "Scheduling linen deliveries in a large hospital," *Journal of the Operational Research Society*, pp. 777–780, 1998.

[11] M. W. Carter, J. Farvolden, G. Laporte, and J. Xu, "Solving an integrated logistics problem arising in grocery distribution," 1995.

[12] J. Pacheco, A. Alvarez, I. García, and F. Angel-Bello, "Optimizing vehicle routes in a bakery company allowing flexibility in delivery dates," *Journal of the Operational Research Society*, vol. 63, no. 5, pp. 569–581, 2012.

[13] V. Hemmelmayr, K. F. Doerner, R. F. Hartl, and M. W. Savelsbergh, "Delivery strategies for blood products supplies," *OR spectrum*, vol. 31, no. 4, pp. 707–725, 2009.

[14] I. Chao, B. L. Golden, E. Wasil *et al.*, "An improved heuristic for the period vehicle routing problem," *Networks*, vol. 26, no. 1, pp. 25–44, 1995.

[15] J.-F. Cordeau, M. Gendreau, and G. Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems." *Networks*, no. 30, pp. 105–119, 1997.

[16] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl, "A variable neighborhood search heuristic for periodic routing problems," *European Journal of Operational Research*, vol. 195, no. 3, pp. 791–802, 2009.

[17] V. Cacchiani, V. Hemmelmayr, and F. Tricoire, "A set-covering based heuristic algorithm for the periodic vehicle routing problem," *Discrete Applied Mathematics*, vol. 163, pp. 53–64, 2014.

[18] R. Baldacci, M. Battarra, and D. Vigo, "Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs," *Networks*, vol. 54, no. 4, pp. 178–189, 2009.

[19] D. d. S. Campos, "Integração dos problemas de carregamento e roteamento de veículos com janela de tempo e frota heterogênea." Ph.D. dissertation, Universidade de São Paulo, 2008.

[20] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated local search*. Springer, 2003.

[21] L. Tang and X. Wang, "Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem," *The International Journal of Advanced Manufacturing Technology*, vol. 29, no. 11-12, pp. 1246–1258, 2006.

[22] J.-F. Cordeau, G. Laporte, A. Mercier *et al.*, "A unified tabu search heuristic for vehicle routing problems with time windows," *Journal of the Operational research society*, vol. 52, no. 8, pp. 928–936, 2001.