Uma heurística baseada em *Iterated Local Search* para o Problema de Roteamento de Veículos com Entregas Fracionárias

Marcos de Melo da Silva[1], **Luiz Satoru Ochi**[1]
[1]Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 3º andar, São Domingos, 24210-240, Niterói, RJ
{mmsilva, satoru}@ic.uff.br

RESUMO

O Problema de Roteamento de Veículos com Entregas Fracionárias (PRVEF) é uma relaxação do Problema de Roteamento de Veículos (PRV) onde os clientes podem ser atendidos por mais de um veículo. O objetivo é determinar um conjunto de rotas de custo mínimo. Neste trabalho é proposto um algoritmo baseado na metaheurística *Iterated Local Search* (ILS), que utiliza o método *Variable Neighborhood Descent* (VND) com ordem aleatória de vizinhanças (RVND) na fase de busca local. A heurística desenvolvida foi testada em três conjuntos de instâncias disponíveis na literatura, compostos de 95 problemas-teste e o número de clientes variando entre 8 e 288. Os resultados obtidos pelo algoritmo, em termos de qualidade das soluções, foram competitivos, sendo capaz de encontrar 65 novas soluções.

PALAVRAS CHAVE. Heurística híbrida, Entregas Fracionárias, *Iterated Local Search*.

ABSTRACT

The Split Delivery Vehicle Routing Problem (SDVRP) is a relaxation of the Vehicle Routing Problem (VRP) where a delivery to a demand point can be split between any number of vehicles. The goal is to minimize the total traveling costs. This work proposes an algorithm based on the metaheuristic Iterated Local Search (ILS) that uses the Variable Neighborhood Descent with Random Neighborhood Ordering (RVND) method in the local search phase. The developed heuristic was tested in three set of instances available in the literature. The results obtained by the algorithm, in terms of solution quality, were competitive, being capable to found 65 new best solutions.

KEY WORDS. Hybrid Heuristic, Split Deliveries, Iterated Local Search.

1 Introdução

Dada uma frota homogênea de veículos localizada em um depósito e um conjunto de clientes geograficamente distribuído com demandas conhecidas de um certo produto, o Problema de Roteamento de Veículos (PRV) consiste em determinar um conjunto de rotas de custo mínimo que inicie e termine no depósito, cada cliente tenha toda sua demanda atendida em uma única visita e para cada rota, a soma das demandas dos clientes atendidos não exceda a capacidade do veículo associado. O Problema de Roteamento de Veículos com Entregas Fracionárias (PRVEF) resulta da relaxação da condição que obriga a cada cliente ter toda sua demanda atendida em uma única visita, ou seja, no PRVEF cada cliente pode ter sua demanda atendida por mais de um veículo. Uma consequência direta disto é que a demanda dos clientes pode exceder a capacidade do veículo. Na literatura o PRVEF é também conhecido como *Split Delivery Vehicle Routing Problem* (SDVRP).

O PRVEF foi introduzido na literatura por Dror e Trudeau (1989, 1990). Os autores mostraram que apesar do PRVEF ser uma relaxação do PRV, continua sendo um problema *NP-difícil*. Por meio de um estudo empírico mostraram também que as economias obtidas ao permitir que um cliente seja atendido por mais de um veículo, tanto em relação ao número de veículos utilizados quanto em relação a distância total percorrida, são significativas.

Alguns autores aplicaram o PRVEF em situações práticas. Em Sierksma e Tijssen (1998), os autores modelaram o problema de escalonamento de helicópteros para o transporte de passageiros entre a base e plataformas petrolíferas no Mar do Norte como PRVEF e utilizaram algoritmos heurísticos e de geração de colunas na resolução. Já o trabalho de Mullaseril *et al*. (1997) descreveu um problema de distribuição de suprimentos em uma fazenda no Arizona para o qual os autores adaptaram a heurística de Dror e Trudeau (1989) para tratá-lo. O trabalho de Song *et al*. (2002) também utilizou uma abordagem de entregas fracionárias para a alocação de produção e roteamento dos veículos que realizam entregas de jornais.

Vários autores desenvolveram abordagens exatas para resolver o PRVEF. Dror *et al*. (1994) desenvolveram uma formulação de programação inteira, propuseram várias desigualdades válidas e um algoritmo de *Branch-and-Bound*. Belenguer *et al*. (2000) desenvolveram um algoritmo de plano de cortes para obter limites inferiores para o problema. Moreno *et al*. (2010) utilizaram uma formulação estendida para desenvolver várias classes de desigualdades válidas que foram incorporadas em um algoritmo de plano de cortes. Archetti *et al*. (2011) desenvolveram um algoritmo *Branch-and-Price-and-Cut*.

Várias abordagens heurísticas foram desenvolvidas para o PRVEF. A primeira heurística, uma busca local, foi desenvolvida por Dror e Trudeau (1989). Archetti *et al*. (2006) propuseram uma heurística baseada em Busca Tabu. Boudia *et al*. (2007) desenvolveram um algoritmo memético com gerência de população. Uma heurística baseada em *Scatter Search* foi proposta por Campos *et al*. (2008). Um algoritmo construtivo iterativo que utiliza o VND como busca local foi desenvolvido por Aleman *et al*. (2010). Para uma revisão abrangente do PRVEF, suas variantes e métodos de resolução ver Archetti e Speranza (2012).

Este trabalho propõe um algoritmo baseado na metaheurística *Iterated Local Search* (ILS) (Lourenço *et al*., 2010), utilizando o *Variable Neighborhood Descent* (VND) (Hansen *et al*., 2010) com ordem aleatória de vizinhanças (RVND) (Subramanian *et al*., 2010) como método de busca local para tratar o PRVEF com frota limitada.

O restante do trabalho está organizado da seguinte forma. A Seção 2 apresenta a definição do problema. A Seção 3 descreve o algoritmo proposto. Os resultados computacionais estão reportados na Seção 4. As considerações finais são feitas na Seção 5.

2 Definição do Problema

Dado um grafo $G = (V, E)$, $V = \{0, 1, \ldots, n\}$ é um conjunto de (n + 1) vértices sendo o depósito representado pelo vértice 0, onde uma frota homogênea de K veículos com capacidade Q está posicionada e os clientes representados pelos vértices $N = \{1, \ldots, n\}$, cada um com uma demanda d_i $(i \in V, d_0 = 0)$ positiva e $E = \{(i, j) : i, j \in V, i < j\}$ é o conjunto de arestas. Um custo não negativo c_{ij} está associado a cada aresta $(i, j) \in E$ e representa a distância para ir de i até j. O objetivo do PRVEF é encontrar um conjunto rotas de custo mínimo onde, cada rota inicie e termine no depósito, a demanda de todos os clientes devem ser satisfeitas e a somas das demandas dos clientes atendidos em cada rota não pode exceder a capacidade Q do veículo associado. A demanda de um cliente pode ser atendida por mais de um veículo (Dror *et al*., 1994).

Na literatura, duas versões do problema são considerados. Na primeira uma frota ilimitada de veículos está disponível e na segunda, a frota é limitada em m veículos. Como no PRVEF é possível atender todos os clientes com uma frota mínima $K_{min} = \lceil (\sum_{i=1}^{n} d_i)/Q \rceil$, o número de veículos na segunda versão do problema é definido como $m = K_{min}$. Neste trabalho tratamos a segunda versão do problema.

Em Dror e Trudeau (1989), os autores mostraram que nas instâncias em que a demanda média dos clientes é maior que 10% da capacidade do veículo, os ganhos ao permitir que os clientes sejam atendidos por mais de um veículo são significativos quando comparado com a solução obtida pelo VRP clássico.

Archetti *et al*. (2008) conduziram um estudo mais acurado dos benefícios obtidos ao se permitir divisão nas entregas. Observou-se que a redução máxima que pode ser obtida no número de veículos é de 50%. Além disso, os maiores ganhos ocorrem quando a demanda média dos clientes está entre 50% e 75% da capacidade do veículo e a variância é pequena. Verificou-se ainda que não existe dependência aparente entre a redução dos custos de distribuição e a localização dos clientes.

3 Algoritmo proposto

O algoritmo proposto para o PRVEF, denominado SPLITILS, é uma heurística *multi-start* baseada na metaheurística ILS e que utiliza o procedimento VND com ordem aleatória na escolha das vizinhanças (RVND) na fase de busca local. O pseudocódigo é mostrado no Algoritmo 1. No laço mais externo (linhas 3 a 24), a heurística executa $maxIter$ iterações e em cada uma delas uma solução inicial é gerada por um procedimento baseado no algoritmo de Clarke e Wright (1964) (linha 5). Em seguida, o ILS (linhas 8 a 19) tenta melhorar a solução gerada por meio de uma busca local utilizando o RVND (linha 9). Após a busca local, se a solução s contiver mais rotas que o número mínimo de veículos, as rotas de menor carga são esvaziadas (linhas 10 a 12). Quando uma solução de melhor qualidade é encontrada o contador de iterações do ILS (*iterILS*) é reiniciado (linhas 13 a 16). Após essa verificação a solução é perturbada (linha 17) e o contador de iterações é incrementado. Como critério de parada é utilizado o número de perturbações sem melhora ($maxIterILS$). A perturbação é sempre realizada sobre a melhor solução corrente s'. O algoritmo retorna a melhor solução s^* encontrada entre todas as iterações.

3.1 Construção de soluções iniciais

Uma solução inicial para PRVEF é gerada por um procedimento construtivo baseado no algoritmo de Clarke e Wright (1964) adaptado para para aceitar divisão nas demandas. O pseudocódigo do procedimento é mostrado no Algoritmo 2. O algoritmo constrói inicialmente uma solução s com n rotas que visitam diretamente cada um dos clientes (linha 3) e calcula para cada duas rotas, a economia gerada ao juntá-las (linha 4). Após ordenar a lista de economias LS (linha 5), tenta-se juntar duas rotas enquanto houver custos positivos e a soma da carga das duas rotas

Algoritmo 1: SPLITILS

```
1  Procedimento SPLITILS(maxIter, maxIterILS, alphaRange)
2  f* ← ∞;
3  para i ← 1, ..., maxIter faça
4      selecione α aleatoriamente do intervalo alfaRange;
5      s ← ConstrucaoCW(α);
6      s' ← s;
7      iterILS ← 0;
8      enquanto iterILS < maxIterILS faça
9          s ← RVND(s);
10         enquanto não houver o número mínimo de rotas faça
11             Esvazie a rota de menor carga redistribuindo seus clientes entre as outras rotas;
12         fim enqto
13         se f(s) < f(s') então
14             s' ← s;
15             iterILS ← 0;
16         fim se
17         s ← Perturba(s');
18         iterILS ← iterILS + 1;
19     fim enqto
20     se f(s') < f* então
21         s* ← s';
22         f* ← f(s');
23     fim se
24 fim para
25 retorna s*;
26 fim SPLITILS
```

não exceder a capacidade do veículo (linhas 6 a 10). Se na solução gerada houver mais rotas que o número mínimo necessário, tenta-se esvaziar rotas redistribuindo os clientes, dividindo a demanda de um cliente entre mais de uma rota se não for possível atendê-lo em com uma única visita (linhas 11 a 13).

Algoritmo 2: ConstrucaoCW

```
1  Procedimento ConstrucaoCW(α)
2  s ← ∅;
3  Inicialize s com rotas diretas para cada cliente i ∈ V − {0}: s ∪ {i};
4  Inicialize LS com as economias geradas ao juntar cada rota i, j ∈ s:
   LS_ij ← c_0i + c_0j − (α × c_ij);
5  Ordene a lista LS em ordem decrescente;
6  enquanto (LS ≠ ∅) e (LS_ij > 0) faça
7      se não exceder a capacidade do veículo e for possível então
8          Junte as rotas i, j ∈ s em uma única rota;
9      fim se
10 fim enqto
11 enquanto não houver o número mínimo de rotas faça
12     Esvazie a rota de menor carga redistribuindo seus clientes entre as outras rotas;
13 fim enqto
14 retorna s;
15 fim ConstrucaoCW
```

3.2 Busca local

A fase de busca local é realizada por um método baseado no procedimento VND com escolha aleatória da ordem em que as estruturas de vizinhanças serão aplicadas (RVND). Seja t o número de estruturas vizinhanças e $N = \{N^1, N^2, N^3, \ldots, N^t\}$, o RVND selecionará aleatoriamente a ordem em que estas estruturas serão executadas. N será composto apenas das vizinhanças inter-rotas. Testes empíricos mostraram que esta abordagem, em média, encontra resultados de melhor qualidade quando comparado com versões que utilizam ordem determinística (Penna *et al*., 2011).

O pseudocódigo do procedimento RVND é apresentado no Algoritmo 3. Primeiramente, a Lista de Vizinhanças (LV) é inicializada com as estruturas de vizinhança inter-rotas (linha 2). A cada execução do laço principal do método (linhas 3 a 14), uma estrutura de vizinhança $N^{(\eta)} \in LV$ é selecionada aleatoriamente (linha 4) e então o melhor vizinho viável encontrado ao aplicar esta vizinhança é armazenado em s' (linha 5). Em caso de melhora, uma busca local intra-rota é realizada nas rotas modificadas (linha 8) e LV é reinicializada com todas as estruturas de vizinhança (linhas 6 a 11). Caso contrário, $N^{(\eta)}$ é removida de LV (linha 12). O procedimento termina quando LV estiver vazia.

Algoritmo 3: RVND

```
1  Procedimento RVND(s)
2  Inicializar lista de vizinhança LV;
3  enquanto LV ≠ ∅ faça
4      Selecione aleatoriamente a vizinhança N^(η) ∈ LV;
5      Encontre o melhor vizinho s' de s ∈ N^(η);
6      se f(s') < f(s) então
7          s ← s';
8          s ← BuscaIntraRota(s);
9          f(s) ← f(s');
10         Atualiza LV;
11     senão
12         Remova N^(η) de LV;
13     fim se
14 fim enqto
15 retorna s;
16 fim RVND
```

Um conjunto de dez estruturas de vizinhança inter-rotas foi utilizado. Destas, cinco são baseadas no esquema λ-*interchanges* proposto por Osman (1993), que consiste em permutar λ clientes entre duas rotas. Foi utilizado $\lambda = 2$ para limitar o número de possibilidades. Das outras cinco estruturas, uma é baseada no operador *Cross-exchange* de Taillard *et al*. (1997) e consiste em permutar dois segmentos de rotas diferentes, uma outra estrutura é baseada no procedimento *RouteAdition*, específico para o PVREF, proposto por Dror e Trudeau (1989) e as últimas três estruturas foram propostas por Boudia *et al*. (2007) e são específicas para o PVREF, uma delas é uma generalização do procedimento *k-Split* proposto por Dror e Trudeau (1989).

Cada uma das estruturas de vizinhança inter-rota do conjunto N é descrita a seguir:

- ***Shift(1,0)*** $- N^{(1)} -$ Um cliente i é transferido da rota r_1 para a rota r_2.
- ***Swap(1,1)*** $- N^{(2)} -$ Permuta o cliente i da rota r_1 com o cliente j da rota r_2.

- ***Shift(2,0)*** $-$ $N^{(3)}$ $-$ Dois clientes adjacentes, i e $i+1$, são transferidos da rota r_1 para a rota r_2.
- ***Swap(2,1)*** $-$ $N^{(4)}$ $-$ Permuta dois clientes adjacentes, i e $i+1$, da rota r_1 com o cliente j da rota r_2.
- ***Swap(2,2)*** $-$ $N^{(5)}$ $-$ Permuta dois clientes adjacentes, i e $i+1$, da rota r_1 com dois clientes adjacentes, j e $j+1$, da rota r_2.
- ***Cross*** $-$ $N^{(6)}$ $-$ O arco entre dois clientes adjacentes, i e $i+1$, da rota r_1 e o arco entre dois clientes adjacentes, j e $j+1$, da rota r_2 são removidos. Em seguida são criados dois arcos, um conectando i e j e outro conectando $i+1$ e $j+1$.
- ***Swap*(1,1)*** $-$ $N^{(7)}$ $-$ Se a demanda do cliente i da rota r_1 for maior que a demanda do cliente j da rota r_2, permuta-se parte da demanda do cliente i da rota r_1 com toda a demanda do cliente j da rota r_2. O movimento é análogo se a demanda do cliente i da rota r_1 for menor que a demanda do cliente j da rota r_2. A carga em ambas as rotas não é alterada.
- ***Swap*(2,1)*** $-$ $N^{(8)}$ $-$ Se a demanda do cliente i da rota r_1 for maior que a demanda dos clientes adjacentes j e $j+1$ da rota r_2, permuta-se parte da demanda do cliente i da rota r_1 com toda a demanda dos cliente j e $j+1$ da rota r_2. A carga em ambas as rotas não é alterada.
- ***RouteAdition*** $-$ $N^{(9)}$ $-$ Dado um cliente i que aparece em pelo menos duas rotas, r_1 e r_2, remove o cliente i de ambas as rotas e cria uma nova.
- ***k-Split*** $-$ $N^{(10)}$ $-$ Remove o cliente i de todas as rotas em que ele aparece. Encontra a melhor posição de inserção de i em cada rota que tenha capacidade residual para atender parte de sua demanda. Resolve o problema da mochila para determinar quais rotas serão utilizadas para atender i.

Se após a aplicação de um desses operadores as partes da demanda de um cliente são movidos para a mesma rota, todas estas estruturas foram adaptadas para juntar estas partes em uma só, evitando assim que um cliente tenha mais de uma cópia em uma mesma rota.

Cada uma destas estruturas é examinada exaustivamente e somente o movimento de melhora mais significativo para cada vizinhança é considerado. Somente movimentos viáveis são considerados, ou seja, apenas as soluções vizinhas nas quais a capacidade dos veículos não seja violada. A complexidade computacional de cada uma dessas estruturas, com exceção de *RouteAdition* e *k-Split*, são da ordem de $O(n^2)$.

Seja N' o conjunto composto por l estruturas vizinhanças intra-rotas, o Algoritmo 4 descreve o procedimento empregado na busca local intra-rota. Inicialmente, a Lista de Vizinhanças (LN) é inicializada com as estruturas de vizinhança intra-rotas (linha 2). Em seguida uma estrutura de vizinhança $N'^{(\nu)} \in LN$ é selecionada aleatoriamente (linha 4) e uma busca local exaustiva é realizada enquanto LN não estiver vazia (linhas 3 a 13).

O conjunto N' é composto de cinco estruturas de vizinhança intra-rota amplamente exploradas na literatura, a saber:

- ***Exchange*** $-$ $N'^{(1)}$ $-$ Permutação entre dois clientes.
- **2-*opt*** $-$ $N'^{(2)}$ $-$ Dois arcos não adjacentes são removidos e outros dois são adicionados formando um novo percurso.
- **Reinserção** $-$ $N'^{(3)}$ $-$ Um único cliente é removido e inserido em outra posição do percurso.

Algoritmo 4: BuscaIntraRota

```
1  Procedimento BuscaIntraRota(s)
2  Inicializar lista de vizinhança LN;
3  enquanto LN ≠ ∅ faça
4      Selecione aleatoriamente a vizinhança N'^(ν) ∈ LN;
5      Encontre o melhor vizinho s' de s ∈ N'^(ν);
6      se f(s') < f(s) então
7          s ← s';
8          f(s) ← f(s');
9          Atualiza LN;
10     senão
11         Remova N^(ν) de LN;
12     fim se
13 fim enqto
14 retorna s;
15 fim BuscaIntraRota
```

- **Or**-*opt2* – $N'^{(4)}$ – Dois clientes adjacentes são removidos e inseridos em outra posição do percurso.
- **Or**-*opt3* – $N'^{(5)}$ – Três clientes adjacentes são removidos e inseridos em outra posição do percurso.

Cada uma destas estruturas é examinada exaustivamente e somente o movimento de melhora mais significativo para cada vizinhança é considerado.

3.3 Mecanismos de perturbação

Toda vez que o procedimento $Perturba()$ é chamado, um movimento é selecionado aleatoriamente do conjunto P composto por três mecanismos de perturbação. Cada um destes mecanismos é descrito a seguir:

- ***PRouteAdition*** – $P^{(1)}$ – Baseado na estrutura *RouteAdition*. Um cliente que apareça em duas ou mais rotas é selecionado aleatoriamente. O cliente é removido de todas as rotas em que ele aparece e uma nova rota que contém este cliente é criada. Caso nenhum cliente apareça em mais de uma rota, outro mecanismo de perturbação é selecionado.
- ***Multiple-Split*** – $P^{(2)}$ – O movimento *k-Split* é aplicado múltiplas vezes e a cada vez um cliente i diferente é aleatoriamente selecionado. Após testes preliminares, o número de vezes que o movimento será aplicado é selecionado do intervalo $[2, 3, 4]$.
- ***Multiple-Swap*(1,1)*** – $P^{(3)}$ – O movimento *Swap*(1,1)* é aplicado múltiplas vezes aleatoriamente. Após testes preliminares, o número de vezes que o movimento é aplicado foi setado em cinco.

4 Resultados computacionais

O algoritmo SPLITILS foi implementado na linguagem C++ (g++ 4.4.3) e executado em um computador com processador Intel® Core™ i7 com 2.93 GHz, 8.0 GB de memória RAM e sistema operacional GNU/Linux Ubuntu 10.04 (*kernel* 2.6.32-25). Foi utilizado uma única *thread*.

O número de iterações ($maxIter$) foi setado em 10. Quanto ao número de perturbações ($maxIterILS$), dois critérios foram empregados, sendo considerado o que ocorrer primeiro. Seja $\beta = \sum_{i=1}^{n} d_i/Q(n-1)$ a fração de utilização da capacidade do veículo, no primeiro critério, $maxIterILS = n \times v$ se $\beta < 0.25$, $maxIterILS = \lceil (n \times v)/2 \rceil$ se $0.25 \leq \beta < 0.5$, $maxIterILS = \lceil (n \times v)/3 \rceil$ se $0.5 \leq \beta < 0.75$ e $maxIterILS = \lceil (n \times v)/4 \rceil$ se $\beta \geq 0.75$. O segundo critério é $maxIterILS = 30seg$. O intervalo $alphaRange$ foi setado como $alphaRange = [0.1, 0.2, \ldots, 2)$. Estes valores foram calibrados empiricamente por meio de testes preliminares. Para cada instância foram realizadas 30 execuções do SPLITILS.

Nas tabelas apresentadas a seguir, **Problema** indica o nome da instância, **Best Sol.** a melhor solução encontrada pelo respectivo trabalho, **Avg. Gap** o *gap* entre a solução média obtida pelo SPLITILS e a melhor solução da literatura utilizando a fórmula $gap = (100 \times (SPLITILS - Literatura)/Literatura)$ (*gaps* negativos indicam que as soluções encontradas pelo SPLITILS possuem um custo menor que o da literatura), **Avg. Sol.** a média das soluções obtidas, **Avg. Time** a média dos tempos em segundos das 30 execuções do SPLITILS.

Tabela 1: Resultados computacionais para as instâncias de Belenguer *et al.* (2000)

	Literatura	SPLITILS			
Problema	Best Sol.	Best Sol.	Avg. Sol.	Avg. Gap	Avg. Time
eil22	*375[a]	375	375,00	0,00	0,18
eil23	*569[a]	569	569,00	0,00	0,14
eil30	503[b]	503	503,00	0,00	0,29
eil33	*835[a]	835	835,00	0,00	0,56
eil51	*521[a]	521	521,00	0,00	2,45
eilA76	828[b]	818	821,33	-0,81	18,71
eilB76	1019[b]	1002	1006,63	-1,21	28,62
eilC76	735[a]	733	733,60	-0,19	12,34
eilD76	682[b]	681	682,77	0,11	10,32
eilA101	817[a]	814	815,23	-0,22	26,78
eilB101	1077[a]	1061	1064,70	-1,14	56,53
S51D1	458[a]	458	458,00	0,00	1,42
S51D2	707[b]	703	704,90	-0,30	5,47
S51D3	945[b]	943	944,53	-0,05	5,54
S51D4	1578[b]	1552	1556,83	-1,34	12,85
S51D5	1351[b]	1328	1331,03	-1,48	12,92
S51D6	2182[b]	2160	2167,23	-0,68	11,72
S76D1	592[b]	592	592,37	0,06	6,41
S76D2	1089[b]	1082	1083,93	-0,47	35,34
S76D3	1427[b]	1420	1424,40	-0,18	26,35
S76D4	2117[b]	2072	2075,90	-1,94	64,34
S101D1	716[a]	716	717,07	0,15	20,49
S101D2	1372[b]	1367	1371,60	-0,03	88,08
S101D3	1891[b]	1867	1872,30	-0,99	94,66
S101D5	2854[b]	2774	2787,17	-2,34	210,71

a - Belenguer *et al.* (2000); b - Boudia *et al.* (2007): Pentium IV 3.0 GHz
* - Solução ótima provada por Belenguer *et al.* (2000)

O SPLITILS foi testado em três conjuntos de instâncias. O primeiro foi gerado por Belenguer *et al.* (2000) e é composto de 25 problemas-testes dos quais, 11 foram selecionados da TSPLIB (Reinelt, 1991) e 14 foram gerados aleatoriamente da seguinte forma. As coordenadas dos clientes correspondem às das instâncias *eil51*, *eil76* e *eil101* da TSPLIB. A capacidade dos veículos é sempre $Q = 160$ e as demandas dos clientes foram geradas conforme Dror e Trudeau

(1989). O número que aparece no nome da instância indica a quantidade de clientes mais o depósito. Nas 14 instâncias geradas aleatoriamente, D1 significa que a demanda de cada cliente foi selecionada do intervalo $d_i \in [\lceil 0.01Q \rceil, \lfloor 0.1Q \rfloor]$, D2 do intervalo $d_i \in [\lceil 0.1Q \rceil, \lfloor 0.3Q \rfloor]$, D3 do intervalo $d_i \in [\lceil 0.1Q \rceil, \lfloor 0.5Q \rfloor]$, D4 do intervalo $d_i \in [\lceil 0.1Q \rceil, \lfloor 0.9Q \rfloor]$, D5 do intervalo $d_i \in [\lceil 0.3Q \rceil, \lfloor 0.5Q \rfloor]$ e D6 do intervalo $d_i \in [\lceil 0.7Q \rceil, \lfloor 0.9Q \rfloor]$.

O segundo conjunto de instâncias foi gerado por Campos *et al*. (2008) com as instâncias utilizadas no trabalho de Archetti *et al*. (2006). O conjunto é composto de 49 problemas-teste com até 199 clientes e dividido em sete grupos, sendo o primeiro composto das instâncias com demandas originais e os outros seis com suas demandas geradas conforme os intervalos do primeiro conjunto.

O terceiro conjunto de instâncias foi gerado por Chen *et al*. (2007). O conjunto é composto de 21 problemas-teste variando de 8 a 288 clientes, a capacidade de todos os veículos é $Q = 100$ e as demandas de cada um dos clientes foi setado como 60 ou como 90. Além disso, todos os problemas possuem simetria geométrica com os clientes localizados em círculos concêntricos em torno do depósito. Este tipo de posicionamento dos clientes permite que soluções próximas do ótimo sejam estimadas visualmente.

Tabela 2: Resultados computacionais para as instâncias de Chen *et al*. (2007)

	Literatura	SPLITILS			
Problema	Best Sol.	Best Sol.	Avg. Sol.	Avg. Gap	Avg. Time
SD1	*228,28[b]	228,28	228,28	0,00	0,01
SD2	*708,28[b]	708,28	708,28	0,00	0,13
SD3	*430,58[b]	430,58	430,58	0,00	0,13
SD4	*631,05[b]	631,05	631,05	0,00	0,49
SD5	1390,57[a]	1390,57	1390,57	0,00	1,28
SD6	*831,24[b]	831,24	831,24	0,00	1,27
SD7	3640,00[a]	3640,00	3640,00	0,00	2,90
SD8	5068,28[a]	5068,28	5068,28	0,00	6,97
SD9	*2044,20[b]	2044,20	2047,25	0,15	7,25
SD10	*2684,90[b]	2684,88	2688,44	0,13	22,62
SD11	13280,00[a]	13280,00	13280,00	0,00	37,86
SD12	7279,97[a]	7216,11	7219,14	-0,84	45,19
SD13	10110,58[a]	10110,58	10112,88	0,02	74,53
SD14	10893,50[a]	10717,52	10725,74	-1,54	209,78
SD15	15168,28[a]	15096,78	15104,17	-0,42	287,34
SD16	*3381,30[b]	3409,00	3410,67	0,87	298,02
SD17	26559,93[a]	26496,08	26500,92	-0,22	300,66
SD18	14440,59[a]	14202,52	14210,73	-1,59	300,67
SD19	20191,19[a]	19996,98	20021,24	-0,84	301,07
SD20	39813,49[a]	39645,84	39659,21	-0,39	302,01
SD21	*11271,00[b]	11395,57	11409,89	1,23	303,41

a - Aleman *et al*. (2010): Pentium IV 2.8 GHz; b - Moreno *et al*. (2010): Core 2 Duo, 2.0 GHz
* - Solução ótima provada por Moreno *et al*. (2010)

Na Tabela 1 estão representados os resultados obtidos pelo SPLITILS para as instâncias de Belenguer *et al*. (2000). O custo da solução foi calculado com arredondamento para o inteiro mais próximo. As melhores soluções para estas instâncias utilizando esta forma de cálculo de custo são encontradas nos trabalhos de Belenguer *et al*. (2000) e Boudia *et al*. (2007). Neste conjunto de instâncias o SPLITILS foi capaz de encontrar 16 novas soluções. Em média o *gap* foi de -0.52. Boudia *et al*. (2007) utiliza um veículo a mais que o número mínimo na instância *eil30*, assim, para esta instância também utilizamos um veículo a mais.

Tabela 3: Resultados computacionais para as instâncias de Campos *et al.* (2008)

	Literatura	SPLITILS			
Problema	Best Sol.	Best Sol.	Avg. Sol.	Avg. Gap	Avg. Time
p1-50	524,6111[b]	524,6111	524,6111	0,00	2,56
p2-75	823,8833[b]	823,8883	824,6775	0,10	21,09
p3-100	829,4397[b]	826,1370	826,3881	-0,37	32,00
p4-150	1042,3740[b]	1025,2360	1027,0411	-1,47	167,13
p5-199	1311,5937[b]	1288,8595	1299,9753	-0,89	303,32
p6-120	1041,2000[b]	1037,8753	1040,3090	-0,09	82,56
p7-100	819,5575[b]	819,5575	819,5575	0,00	22,25
p1-50D1	460,7896[b]	460,7896	460,7896	0,00	1,59
p2-75D1	596,9872[a]	596,2499	596,2499	-0,12	6,79
p3-100D1	726,8076[b]	726,8076	729,5859	0,38	26,08
p4-150D1	871,2624[a]	866,3116	866,4435	-0,55	98,12
p5-199D1	1018,7055[b]	1017,2789	1018,5254	-0,02	273,71
p6-120D1	976,5688[b]	975,9551	976,4052	-0,02	48,83
p7-100D1	635,9953[a]	632,6287	632,9311	-0,48	24,55
p1-50D2	741,0565[a]	741,0562	741,7707	0,10	6,50
p2-75D2	1071,8694[a]	1064,4900	1067,1496	-0,44	39,78
p3-100D2	1392,8469[b]	1374,1473	1379,8455	-0,93	132,85
p4-150D2	1878,7090[b]	1862,1042	1869,8605	-0,47	301,04
p5-199D2	2340,1362[b]	2309,6859	2322,9893	-0,73	303,09
p6-120D2	2720,3752[b]	2705,3531	2710,3689	-0,37	291,05
p7-100D2	1417,2757[b]	1413,8484	1414,1005	-0,22	104,22
p1-50D3	988,3062[b]	982,7663	985,2457	-0,31	6,70
p2-75D3	1413,7965[b]	1393,1129	1393,4695	-1,44	41,05
p3-100D3	1845,2995[b]	1824,5075	1829,7856	-0,84	126,12
p4-150D3	2561,6472[b]	2530,9487	2536,8842	-0,97	301,15
p5-199D3	3191,2475[b]	3176,5936	3187,5092	-0,12	302,90
p6-120D3	3934,3880[b]	3907,8288	3911,9143	-0,57	297,88
p7-100D3	1994,5947[b]	1967,4144	1969,5004	-1,26	120,07
p1-50D4	1467,0635[b]	1455,9973	1459,4372	-0,52	16,98
p2-75D4	2102,5762[b]	2082,8864	2085,7536	-0,80	85,93
p3-100D4	2780,9492[b]	2752,3232	2757,5374	-0,84	232,87
p4-150D4	4045,8715[b]	3996,6149	4008,3829	-0,93	300,74
p5-199D4	4941,2170[b]	4864,6353	4878,4593	-1,27	302,50
p6-120D4	6318,3734[b]	6198,6709	6240,8854	-1,23	300,49
p7-100D4	3113,7187[b]	3088,2303	3089,4095	-0,78	200,74
p1-50D5	1477,0135[b]	1467,4718	1467,4732	-0,65	20,28
p2-75D5	2132,1601[b]	2110,5190	2117,6716	-0,68	104,55
p3-100D5	2858,8684[b]	2812,0700	2822,9769	-1,26	246,59
p4-150D5	4045,8715[b]	3998,0402	4009,9408	-0,89	300,80
p5-199D5	5155,3604[b]	5086,6091	5096,9443	-1,13	302,36
p6-120D5	6424,7101[b]	6379,3393	6384,8624	-0,62	300,50
p7-100D5	3155,6915[b]	3126,2859	3127,4746	-0,89	241,40
p1-50D6	2154,3510[b]	2154,6353	2161,0814	0,31	18,10
p2-75D6	3200,3533[b]	3178,7679	3185,7391	-0,46	81,11
p3-100D6	4312,9461[b]	4297,3288	4303,1154	-0,23	218,09
p4-150D6	6267,4827[b]	6233,0904	6240,8765	-0,42	300,43
p5-199D6	8081,5768[b]	8032,9301	8045,8773	-0,44	300,92
p6-120D6	10063,4739[b]	10004,1047	10008,4859	-0,55	300,27
p7-100D6	4919,4826[b]	4904,1854	4907,2212	-0,25	230,16

a - Campos *et al.* (2008): Pentium IV 2.4 GHz; b - Boudia *et al.* (2007): Pentium IV 3.0 GHz

A Tabela 2 apresenta os resultados obtidos pelo SPLITILS para as instâncias de Chen *et al*. (2007). As melhores soluções para estas instâncias são encontradas nos trabalhos de Aleman *et al*. (2010) e Moreno *et al*. (2010). Neste conjunto de instâncias o SPLITILS encontrou a mesma solução da literatura em nove problemas-teste e foi capaz de encontrar outras 7 novas soluções. Em média o *gap* foi de -0.16.

A Tabela 3 apresenta os resultados obtidos pelo SPLITILS para as instâncias de Campos *et al*. (2008). As melhores soluções para estas instâncias são encontradas nos trabalhos de Campos *et al*. (2008) e Boudia *et al*. (2007). Neste conjunto de instâncias o SPLITILS encontrou a mesma solução da literatura em três casos e foi capaz de encontrar outras 42 novas soluções. Em média o *gap* foi de -0.54.

Os tempos computacionais para os trabalhos da literatura não são reportados aqui. Tal decisão foi tomada tendo em vista que parte dos resultados reportados na literatura foram obtidos a partir de métodos exatos que consomem muito tempo computacional. Além disso, uma comparação direta de tempos computacionais não seria possível, visto que diferentes ambientes computacionais foram empregados.

5 Considerações finais

Este trabalho propôs uma heurística baseada em ILS e RVND para o Problema de Roteamento de Veículos com Entregas Fracionárias. Observou-se que a abordagem desenvolvida é simples e sua eficácia, em termos de qualidade das soluções, foi demonstrada por meio de experimentos realizados em três conjuntos de instâncias, compostos por 95 problemas-teste com até 288 clientes. O método apresentado foi capaz de melhorar o resultado de 65 soluções e de chegar à mesma solução conhecida em outras 18 instâncias.

Como trabalhos futuros, pretende-se estender o algoritmo proposto para tratar outras variantes do PRVEF tal como o PRVEF com janelas de tempo e possivelmente testar outros métodos de resolução, como por exemplo abordagens híbridas com programação matemática.

Referências

Aleman, R. E., Zhang, X. e Hill, R. R. (2010), An adaptive memory algorithm for the split delivery vehicle routing problem. *Journal of Heuristics*, v. 16, n. 3, p. 441–473.

Archetti, C., Bianchessi, N. e Speranza, M. G. (2011), A column generation approach for the split delivery vehicle routing problem. *Networks*, v. 58, n. 4, p. 241–254.

Archetti, C. e Speranza, M. G. (2012), Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, v. 19, n. 1-2, p. 3–22.

Archetti, C., Speranza, M. G. e Hertz, A. (2006), A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, v. 40, n. 1, p. 64–73.

Archetti, C., Savelsbergh, M. W. e Speranza, M. G. (2008), To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review*, v. 44, n. 1, p. 114 – 123.

Belenguer, J. M., Martinez, M. C. e Mota, E. (2000), A lower bound for the split delivery vehicle routing problem. *Operations Research*, v. 48, n. 5, p. 801–810.

Boudia, M., Prins, C. e Reghioui, M. An effective memetic algorithm with population management for the split delivery vehicle routing problem. *HM'07: Proceedings of the 4th*

international conference on Hybrid metaheuristics, p. 16–30, Berlin, Heidelberg. Springer-Verlag, 2007.

Campos, V., Corberán, A. e Mota, E. A scatter search algorithm for the split delivery vehicle routing problem. *Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management*, p. 137–152. Springer Berlin, 2008.

Chen, S., Golden, B. e Wasil, E. (2007), The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, v. 49, n. 4, p. 318–329.

Clarke, G. e Wright, J. W. (1964), Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, v. 12, n. 4, p. 568–581.

Dror, M., Laporte, G. e Trudeau, P. (1994), Vehicle routing with split deliveries. *Discrete Applied Mathematics*, v. 50, n. 3, p. 239–254.

Dror, M. e Trudeau, P. (1989), Savings by split delivery routing. *Transportation Science*, v. 23, n. 2, p. 141–145.

Dror, M. e Trudeau, P. (1990), Split delivery routing. *Naval Research Logistics*, v. 37, n. 3, p. 383–402.

Hansen, P., Mladenovic, N., Brimberg, J. e Pérez, J. A. M. Variable neighborhood search. Gendreau, M. e Potvin, J.-Y. (Eds.), *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, p. 61–86. Springer US, 2010.

Lourenço, H., Martin, O. C. e Stützle, T. Iterated local search: Framework and applications. Gendreau, M. e Potvin, J.-Y. (Eds.), *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, p. 363–397. Springer US, 2010.

Moreno, L., de Aragão, M. P. e Uchoa, E. (2010), Improved lower bounds for the split delivery vehicle routing problem. *Operations Research Letters*, v. 38, n. 4, p. 302 – 306.

Mullaseril, P. A., Dror, M. e Leung, J. (1997), Split-delivery routeing heuristics in livestock feed distribution. *The Journal of the Operational Research Society*, v. 48, n. 2, p. 107–116.

Osman, I. H. (1993), Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. v. 41, n. 4, p. 421–451.

Penna, P. H. V., Subramanian, A. e Ochi, L. S. (2011), An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, v. , p. 1–32.

Reinelt, G. (1991), TSPLIB–A Traveling Salesman Problem Library. *INFORMS Journal on Computing*, v. 3, n. 4, p. 376–384.

Sierksma, G. e Tijssen, G. A. (1998), Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research*, v. 76, n. 0, p. 261–286.

Song, S. H., Lee, K. S. e Kim, G. S. (2002), A practical approach to solving a newspaper logistics problem using a digital map. *Computers and Industrial Engineering*, v. 43, n. 1-2, p. 315 – 330.

Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S. e Farias, R. (2010), A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, v. 37, n. 11, p. 1899–1911.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F. e Potvin, J.-Y. (1997), A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, v. 31, n. 2, p. 170–186.