versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

ALGORITMO DE BUSCA DISPERSA APLICADO AO PROBLEMA CLÁSSICO DE ROTEAMENTO DE VEÍCULOS

Nélida Gladys Maquera Sosa
Programa de Engenharia de Produção / COPPE
Universidade Federal do Rio de Janeiro (UFRJ)
nelidagladys@yahoo.com

Roberto Diéguez Galvão*
Programa de Engenharia de Produção / COPPE
Universidade Federal do Rio de Janeiro (UFRJ)
galvao@pep.ufrj.br

Dan Abensur Gandelman
Programa de Engenharia de Produção / COPPE
Universidade Federal do Rio de Janeiro (UFRJ)
dangandelman@hotmail.com

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 06/2006; aceito em 03/2007 após 1 revisão
Received June 2006; accepted March 2007 after one revision

Resumo

Neste artigo apresentamos heurísticas usando o conceito da meta-heurística Busca Dispersa (BD), desenvolvidas para a solução do Problema de Roteamento de Veículos (PRV) Clássico, detalhando cada uma de suas etapas básicas quando aplicadas ao problema em questão. A Busca Dispersa é um método evolutivo que combina soluções com a finalidade de criar novas soluções de melhor qualidade; ainda que apresente similaridades com os algoritmos genéticos difere dos mesmos em princípios fundamentais. Um aspecto importante da BD é formar soluções com alta qualidade para dirigir a busca a regiões promissoras. Experimentos computacionais foram realizados em quatro conjuntos de dados disponíveis na literatura. Os resultados mostram que a BD é robusta e competitiva em termos de qualidade das soluções obtidas e tempo computacional para o PRV Clássico, para os conjuntos de dados testados.

Palavras-chave: busca dispersa; problema de roteamento de veículos; meta-heurísticas.

Abstract

In this paper we present a *Scatter Search* algorithm designed for the solution of the Classical Vehicle Routing Problem, giving details of each of its basic phases when applied to the routing problem. Scatter Search is an evolutionary meta-heuristic that combines solutions with the objective of obtaining new solutions of higher quality; even though there are similarities between Scatter Search and Genetic Algorithms, the two methods differ in basic principles. An important aspect of Scatter Search is to construct high quality solutions in order to direct the search to promising regions. Computational experiments were conducted using four data sets available in the literature. The results show that Scatter Search is robust and competitive in terms of both the quality of solutions and computational times, for the VRP data sets we used.

Keywords: scatter search; vehicle routing problem; meta-heuristics.

1. Introdução

A importância de sistemas de distribuição torna-se evidente quando se considera o impacto dos respectivos custos na operação das empresas. Existe uma variedade de problemas de decisão nessa área, em diferentes níveis: estratégico, tático e operacional. Decisões sobre a localização de facilidades (fábricas, depósitos, entre outras) são consideradas estratégicas; decisões que determinam por exemplo o tamanho e a composição da frota de veículos de uma empresa podem ser consideradas de nível estratégico/tático; a nível operacional temos por exemplo decisões sobre o *roteamento* e/ou *sequenciamento* de frotas de veículos.

O Problema de Roteamento de Veículos (PRV) é um nome genérico dado a uma classe de problemas de distribuição. Um caso particular do problema geral constitui a chamada *versão clássica* do PRV e pode ser definida da seguinte maneira. Seja $G = (U, A)$ um grafo (direcionado ou não) onde $U = \{u_0, u_1, ..., u_n\}$ é o conjunto de vértices (clientes) e $A = \{(u_i, u_j) : u_i, u_j \in U, i \neq j\}$ é o conjunto de arcos. O vértice u_0 representa o depósito de uma frota homogênea de veículos com capacidade Q por veículo. Cada cliente tem uma demanda q_i e requer um tempo de serviço s_i. Uma matriz de custos de viagem C = [c_{ij}] é definida em A. O número de veículos é pré-determinado (*m* veículos) ou é considerado uma variável de decisão. O PRV clássico (com restrições de capacidade e distância máxima) consiste em designar um conjunto de *m* rotas de entrega e/ou coleta tal que: i) o custo total do conjunto de rotas percorrido pela frota seja minimizado; ii) cada rota se inicie e finalize no depósito; iii) cada cliente tenha sua demanda suprida exatamente por um veículo; iv) a carga total de cada veículo não exceda sua capacidade Q ; e v) o tempo total necessário para completar qualquer rota não exceda um limite pré-especificado *T*, que inclui tempos de viagem entre clientes e tempos de serviço em cada cliente.

O PRV é um problema NP-difícil (Lenstra & Rinnooy Kan, 1981). Devido à sua complexidade computacional, métodos exatos de solução são inviáveis para instâncias de grande porte, o que nos remete à utilização de métodos heurísticos/meta-heurísticos. Existe um grande número de artigos que tratam do problema clássico descrito no parágrafo acima, usando diversas heurísticas e meta-heurísticas, com predominância das meta-heurísticas, mais recentes e apresentando resultados de melhor qualidade. Não é nosso objetivo fazer uma revisão detalhada da bibliografia existente sobre o assunto; uma revisão detalhada (mas evidentemente não atualizada) sobre o PRV pode ser encontrada em Laporte *et alii* (2000). Alguns artigos recentes relacionados ao tema são, por exemplo, Cordeau *et alii* (2002), Baker & Ayechew (2003), Vigo & Toth (2003), Wassan (2006). Toth & Vigo (2002) escreveram um importante livro sobre o PRV, englobando tanto o problema clássico como variantes do mesmo.

Recentemente a *Busca Dispersa* (BD), uma nova metodologia com base em populações, tem demonstrado ser muito efetiva na solução de problemas de otimização discreta. Apesar de a *Busca Dispersa* apresentar algumas semelhanças com os *Algoritmos Genéticos* (AG), ela difere dos mesmos em princípios fundamentais, tais como o uso de estratégias determinísticas ao invés de estratégias aleatórias. Um aspecto importante da BD é a relação existente entre a capacidade do método de dirigir a busca a regiões promissoras e sua eficiência na exploração dessas regiões.

A BD proporciona o uso de estratégias flexíveis, que permitem o desenvolvimento de diferentes algoritmos com distintos graus de complexidade para as diferentes etapas da busca. Os conceitos e princípios fundamentais do método foram propostos na década de 70 por Glover

(1977), com base em estratégias para combinar regras de decisão e restrições. Os trabalhos de Glover (1997), Laguna & Martí (2003) e Martí *et alii* (2006) são referências importantes sobre a BD. A Busca Dispersa tem por base o principio de que, dadas duas soluções, pode-se obter novas soluções mediante a combinação delas, de modo a melhorar as que as originaram.

Neste artigo propomos um algoritmo com base na meta-heurística *Busca Dispersa* para o PRV Clássico, testando-o para diferentes conjuntos de dados. O texto está organizado da seguinte forma. Na Seção 2 apresentamos uma descrição básica da BD. Na Seção 3 descrevemos detalhadamente o algoritmo de BD que propomos para o PRV Clássico. Testes computacionais realizados com diferentes conjuntos de dados são mostrados na Seção 4. Por último, na Seção 5, fazemos algumas considerações finais sobre o trabalho.

2. A Meta-heurística Busca Dispersa

A Busca Dispersa (*Scatter Search* em inglês) é um método evolutivo (baseado em populações) que é muito efetivo na solução de diversos problemas de otimização discreta, por exemplo na solução do problema de ordenação linear, ver Campos *et alii* (2001), na otimização global de funções multimodais, ver Laguna & Martí (2000) e na solução de problemas de roteamento de veículos com janelas de tempo, ver Russel & Chiang (2006). Essa meta-heurística combina soluções pertencentes a um conjunto denominado *conjunto de referência* (*Refset*), com o intuito de capturar informação não contida nas soluções originais. O *Refset* guarda "boas" soluções encontradas durante o processo de busca. Cabe destacar que o significado de "boa" não se restringe apenas à qualidade da solução, mas também a sua *diversidade* em relação a outras soluções deste conjunto. A BD é composta basicamente por 5 etapas (métodos), que descrevemos a seguir de forma resumida. Posteriormente explicaremos detalhadamente cada etapa no contexto de uma meta-heurística desenvolvida visando a solução do PRV Clássico. Apresentamos inicialmente a nomenclatura que será utilizada neste trabalho.

P	Conjunto de soluções obtidas com o método gerador de soluções iniciais;
PSize	Tamanho da população P de soluções iniciais (i.e.,$\|P\|$=*Psize*);
Refset	Conjunto de Referência;
Pool	Conjunto de soluções geradas pelo método da combinação;
b	Tamanho do conjunto de referência (i.e., $\|RefSet\|=b$);
b_1	Tamanho do subconjunto do *Refset* correspondente a soluções de qualidade;
b_2	Tamanho do subconjunto do *Refset* correspondente a soluções com diversidade;
x^i	A $i^{ésima}$ solução do conjunto de referência. x^1 é a melhor solução em *Refset* e x^b a pior;
$d(x,y)$	Distância entre a solução x e a solução y.

As etapas da BD são as seguintes:

1. *Etapa Geradora de Soluções Iniciais*: esta etapa gera um conjunto P com *Psize* soluções, de onde extraímos um subconjunto que se denomina conjunto de referência *Refset*.

2. *Etapa de Melhoria*: consiste de uma busca local para melhorar as *Psize* soluções inicialmente encontradas. Se a solução que está sendo avaliada é *inviável*, o método tenta transformá-la em uma solução viável. Se a solução é *viável*, o método tenta melhorá-la.

3. *Geração do Conjunto de Referência*: nesta etapa é construído ou atualizado o conjunto de referência *Refset*.

3.1 *Construção* – A partir do conjunto *P* se extrai *Refset*, combinando soluções de alta qualidade com soluções com diversidade; as soluções de *Refset* são usadas, durante o processo de busca, para gerar novas soluções mediante a combinação de soluções contidas nesse conjunto.

3.2 *Atualização* – A atualização de *Refset* acontece quando novas soluções que atendem os requisitos para ingressar em *Refset* são encontradas. Assim *Refset* mantém seu tamanho $b=b_1+b_2$ constante, mas as soluções pertencentes ao mesmo melhoram durante o processo de busca.

Para atualizar *Refset* tem-se que considerar o *momento* em que a atualização deve ser realizada (estática ou dinâmica). A atualização se diz *estática* quando a mesma é realizada após testadas todas as possíveis combinações em *Refset*; se diz *dinâmica* quando imediatamente após alguma combinação a solução resultante é tentativamente inserida em *Refset*, sempre que atenda os requisitos necessários para ingressar no conjunto.

4. *Geração de Subconjuntos*: opera no *conjunto de referência*, consistindo em criar diferentes subconjuntos de soluções que serão utilizadas na etapa de combinação.

5. *Combinação de Soluções*: utiliza os subconjuntos de soluções gerados na Etapa 4, combinando as soluções em cada subconjunto com o objetivo de encontrar novas soluções. Esta etapa de combinação é um mecanismo específico para cada problema, uma vez que está diretamente relacionado com a representação da solução.

O algoritmo da Figura 1 mostra as etapas descritas por 1-5 em um esquema básico da *Busca Dispersa* para um problema de minimização (atualização de *Refset* estática).

Passo 1. Inicie com $P \leftarrow \varnothing$. Inicie a *etapa geradora de soluções iniciais* e construa uma solução *y*. Aplique o *método de melhoria* a *y* e encontre a solução melhorada *x*. Se $x \notin P$ então adicione *x* a *P*, caso contrário descarte *x*. Repita este passo até $|P| = Psize$.

Passo 2. Construa o *conjunto de referência* $Refset = \{x^1,...,x^b\}$ com as b_1 melhores soluções de *P* e as b_2 soluções de *P* com maior diversidade em relação às soluções que estão em *Refset*. Ordene essas $b=b_1+b_2$ soluções de forma crescente em relação à função objetivo. Faça o operador lógico *newsolutions* ← *TRUE*.

Enquanto *(newsolutions = TRUE.)*

newsolutions ← *FALSE*.

Passo 3. Mediante a *geração de subconjuntos* gere subconjuntos a partir de *RefSet,* agrupando por exemplo pares de soluções desse conjunto.

Enquanto (existem subconjuntos sem examinar)

Passo 4. Aplique o *método de combinação* às soluções de cada subconjunto gerado no **Passo 3** e guarde as soluções obtidas em uma lista denominada *Pool*.

Passo 5. Aplique o *método de melhoria* a cada solução de *Pool* obtida no **Passo 4**.

Fim Enquanto

Passo 6. Caso possível atualize *Refset* (*b* melhores soluções em $Refset \cup Pool$); Reordene *Refset*.

Se *Refset* atualizado, *newsolutions* ← TRUE.

Fim Enquanto

Figura 1 – Esquema Básico de um Algoritmo de Busca Dispersa.

O algoritmo básico de BD termina quando não existem novos elementos a serem combinados em *Refset*.

3. Uma Aplicação da Busca Dispersa ao PRV Clássico

Nesta seção são apresentadas as diferentes etapas de uma aplicação da BD ao PRV clássico. Para a *etapa geradora de soluções* foram testadas duas heurísticas construtivas, apresentadas a seguir. São apresentados também três métodos de melhoria inter-rotas e um método de melhoria intra-rota para a *etapa de melhoria* da BD. As etapas de *construção (atualização) de Refset*, *geração de subconjuntos* e *combinação de soluções* são finalmente descritos.

A construção de *Refset* se inicia a partir do conjunto *P* de *Psize* soluções geradas após as duas etapas iniciais da busca. As soluções de *Refset* são ordenadas em ordem crescente do valor da função objetivo, isto é, de acordo com sua qualidade. Seleciona-se os pares de soluções que serão combinadas e o processo de combinação de soluções de *Refset* é iniciado. *Refset* é atualizado com novas soluções que forem geradas por esse processo; a atualização de *Refset* pode ser *estática* ou *dinâmica*. O critério de parada para a BD é quando for atingido o valor da solução ótima da instância em consideração (quando disponível na literatura), ou quando *Refset* não for atualizado na última etapa da BD.

3.1 Etapa 1: Geração de Soluções Diversas

As heurísticas desenvolvidas para o PRV Clássico podem ser classificadas em três categorias distintas: heurísticas de inserção, heurísticas de duas fases e heurísticas de melhoria. As heurísticas de inserção constroem gradualmente uma solução pela inserção de clientes de acordo com algum critério pré-estabelecido. Nas heurísticas de duas fases o problema é decomposto em duas fases: agrupamento e roteamento. Pode ser feito inicialmente o agrupamento de clientes, seguido pelo seu roteamento; ou as duas fases podem ser invertidas (primeiro o roteamento, seguido do agrupamento). As heurísticas de melhoria podem ser de melhoria inter-rotas ou intra-rotas.

Apresentamos a seguir dois métodos para gerar as *Psize* soluções iniciais (população inicial) da BD: o primeiro uma adaptação da Heurística de Varredura de Gillett & Miller (1974); o segundo com base na heurística de Beasley (1983). Os resultados produzidos por esses algoritmos são melhorados por meio de diversos procedimentos simples na etapa seguinte da BD.

3.1.1 Algoritmo G & M: Com base na heurística de Gillett e Miller

Esta heurística corresponde a um algoritmo de *agrupamento-roteamento*. Na heurística de varredura com base no algoritmo proposto por Gillett & Miller (1974), os agrupamentos são formados girando uma semi-reta com origem no depósito e incorporando os clientes "varridos" por dita semi-reta na mesma rota até que as restrições do PRV Clássico não sejam satisfeitas. Para este algoritmo cada cliente *i* está dado por suas coordenadas polares (r_i, q_i), em um sistema que tem o depósito como origem. Uma versão simplificada da heurística é definida pelos passos descritos a seguir.

Passo 1: Ordenar os clientes segundo o ângulo polar q que cada cliente faz com a semi-reta. Se dois clientes possuem o mesmo valor de q, o cliente com menor valor de r é selecionado. Selecione um cliente u_i como cliente inicial e faça *i=1, k =1*; $C_1 = \{u_1\}$ é o *cluster de ordem 1*.

Passo 2: Se todos os clientes pertencem a algum *cluster*, ir ao **Passo 4**. Caso contrário ir ao **Passo 3**.

Passo 3: Fazer $i = i+1$ e selecionar o próximo cliente u_i. Se u_i puder ser adicionado a C_k, fazer $C_k = C_k \cup \{u_i\}$. Caso contrário, fazer $k = k+1$ e criar um novo agrupamento $C_k = \{u_i\}$. Ir ao **Passo 2**.

Passo 4: Para cada agrupamento C_k, resolver um Problema do Caixeiro Viajante.

Uma solução viável do PRV é assim construída. O processo se repete *n* vezes (onde *n* é o número de clientes, *n =Psize*), iniciando cada execução do algoritmo por um cliente diferente.

3.1.2 Algoritmo B: Com base na heurística de Beasley

Heurística proposta por Beasley (1983), cuja estratégia é rotear primeiro, agrupar depois. Sem considerar as restrições do PRV, na fase de roteamento constrói-se inicialmente um "tour gigante" iniciado no depósito, que percorre todos os clientes e retorna ao depósito. A este "tour gigante" aplica-se a heurística de melhoria *2-opt*, na qual duas arestas do "tour" são trocadas por outras duas não pertencentes ao mesmo, sempre que o resultado da troca diminui o comprimento do "tour" (ver 3.2.2). Na fase de agrupamento as restrições do PRV são consideradas; o "tour gigante" é dividido em um número de rotas viáveis em que os clientes são designados às rotas conforme sua seqüência no mesmo. Uma solução viável do PRV é assim construída.

3.1.3 Comparação Computacional

Diversas experiências computacionais foram realizadas a fim de comparar o desempenho dos algoritmos construtivos com base nas heurísticas de Gillett e Miller e Beasley. Experimentos foram realizados em dois conjuntos de dados, formados respectivamente pelas 14 instâncias descritas em Christofides *et alii* (1979), e pelas 27 instâncias descritas em Augerat *et alii* (1995). As características dessas instâncias são mostradas nas tabelas A.1 e A.2 do Apêndice, respectivamente.

A Tabela 1 abaixo mostra os resultados obtidos para as duas heurísticas de construção: sem melhoria (linha 1) e depois de aplicada a heurística de melhoria (linha 2) às instâncias. Os procedimentos de melhoria (explicados na seção 3.2) foram propostos por diferentes autores, conforme descrito por exemplo em Toth & Vigo (2002). Todos os dados da Tabela 1 são os Desvios Percentuais Médios em Relação às melhores soluções encontradas na literatura (limite superior ou solução ótima), disponíveis em http://neo.lcc.uma.es/radi-aeb/WebVRP/. Observamos na Tabela 1 que ambas heurísticas apresentam desempenhos semelhantes antes e após o procedimento de melhoria. Optamos por trabalhar com o procedimento com base na heurística de Gillett e Miller.

Tabela 1 – Comparação das heurísticas de Construção.

DPRM	Gillett e Miller	Beasley
Sem melhoria	19,33	19,15
Com melhoria	9,31	9,13

3.2 Etapa 2: Melhoria das Soluções

Nesta seção apresentamos as heurísticas de melhoria utilizadas na Fase 2 de nossa heurística de BD para o PRV clássico. Apresentamos três movimentos inter-rotas e um movimento intra-rota.

3.2.1 Movimentos inter-rotas

Apresentamos a seguir três tipos de movimentos inter-rotas: re-alocação, intercâmbio e cruzamento. Os movimentos de re-alocação e intercâmbio foram definidos em artigos descrevendo heurísticas de "primeira geração" para problemas de roteamento de veículos; ver por exemplo Ball *et alii* [Editors] (1995). O movimento de cruzamento está por exemplo descrito em Montané & Galvão (2006), que o utilizaram na solução do problema de roteamento de veículos com entregas e coletas simultâneas.

Re-alocação

Consiste em remover um cliente de uma das rotas e inseri-lo em uma outra rota. Nas Figuras 2(a) e 2(b) apresentamos o movimento de re-alocação. Para este exemplo o depósito é denominado por 0 e na figura está representado por um quadrado; nestas figuras o cliente *a* é mudado de rota; ou, se desejamos considerar a re-alocação em função de arcos, os arcos *(i,0),(a,b)*e *(a,d)* são substituídos respectivamente pelos arcos *(i,a), (a,0)* e *(b,d)*.

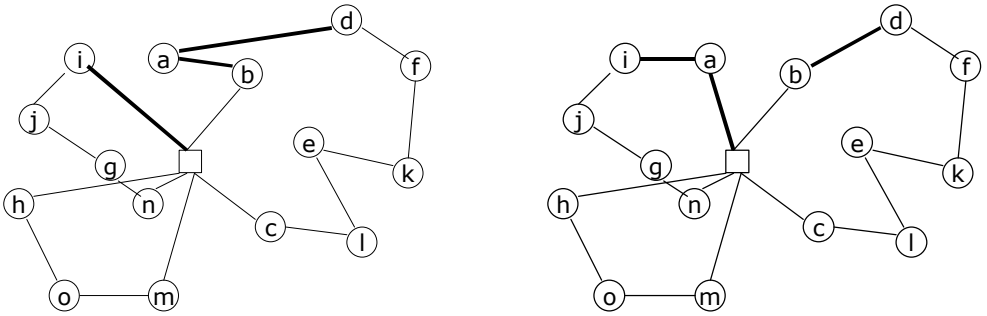


Figura 2 – Movimento de Re-alocação.

Intercâmbio

Consiste em intercambiar clientes entre duas rotas. Nas Figuras 3(a) e 3(b) os clientes *g* e *j* são intercambiados; ou, se desejamos expressar o intercâmbio em função de arcos, temos que os arcos *(i,g), (g,0), (h,j)* e *(j,0)* são substituídos respectivamente pelos arcos *(i,j), (j,0), (h,g)* e *(g,0)*.

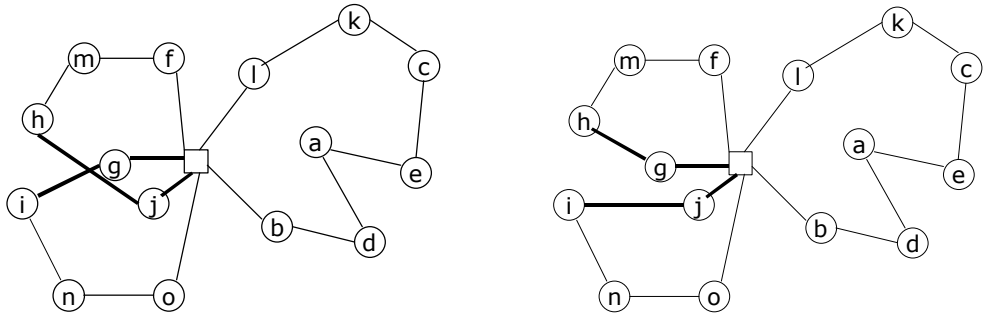


Figura 3 – Movimento de Intercâmbio.

Cruzamento

Neste movimento seleciona-se um par de rotas. Cada rota é dividida em duas seções que são re-conectadas. Consegue-se isso removendo um arco de cada rota e substituindo-os por dois novos arcos que conectam respectivamente a seção inicial da primeira rota com a seção final da segunda rota e a seção inicial da segunda com a seção final da primeira. Na Figura 4 ilustramos este movimento para um par de rotas; a rota p_0-p_{i-1}-p_i-p_0 é substituída pela rota p_0-p_{i-1}-q_j-p_0 e a rota q_0-q_{j-1}-q_j-q_0 é substituída pela rota q_0-q_{j-1}-p_i-q_0.

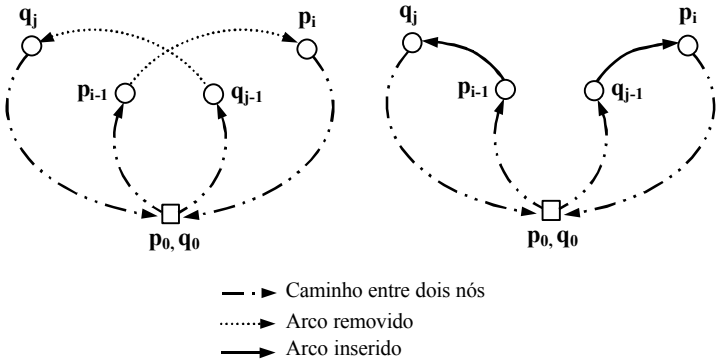


Figura 4 – Movimento de Cruzamento.

3.2.2 Melhoria intra-rotas

Na literatura são encontradas diferentes formas de realizar o movimento intra-rotas; em nosso algoritmo utilizamos o movimento 2-*opt*.

Movimento 2-*opt*

Para a vizinhança intra-rotas utilizamos a heurística *2-opt*, inicialmente proposta por Lin (1965) para o problema do caixeiro viajante; os arcos (*i,o*) e (*b,j*) [Figura 5(a)] são trocados respectivamente pelos arcos (*i,j*) e (*b,o*) [Figura 5(b)].

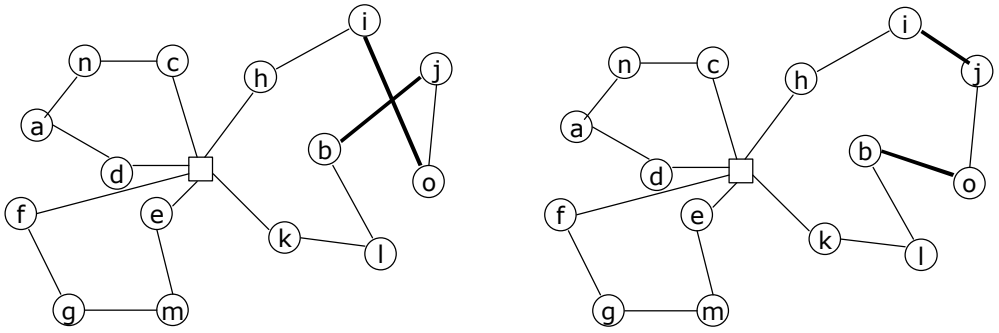


Figura 5 – Movimento 2-*opt*.

Foram aplicadas três variações dos métodos de melhoria às *n* soluções geradas com base no algoritmo de Gilett e Miller: *i*) melhoria quando somente realizam-se movimentos de inserção (ou re-alocação, MMI); *ii*) melhoria quando primeiramente realizam-se movimentos de re-alocação, seguidos de movimentos de intercâmbio e finalmente de movimentos de cruzamento, ou seja, os movimentos são realizados seqüencialmente (MMS) e *iii*) melhoria quando realizam-se os movimentos de re-alocação, intercâmbio e cruzamento separadamente, sendo executado, dentre o melhor movimento de cada uma das três vizinhanças, aquele que proporciona o maior ganho no valor da função objetivo (MMT). Note-se que, em qualquer um dos três casos, (i), (ii) ou (iii), escolhe-se sempre o melhor movimento de cada vizinhança analisada. O movimento 2-*opt* (melhoria intra-rota) é aplicado duas vezes: antes da aplicação dos movimentos inter-rotas e após a aplicação desses movimentos.

Para determinar a melhor estratégia a ser seguida na etapa de melhoria, resultados de experimentos computacionais são apresentados na Tabela 2. O mesmo conjunto das *n* soluções foi sempre utilizado neste experimento. A Tabela 2 mostra, para as três estratégias listadas acima, o desvio percentual médio em relação às melhores soluções encontradas na literatura e o tempo computacional respectivo. Considerando esses resultados decidimos utilizar em nosso algoritmo de BD o método de melhoria MMS.

Tabela 2 – Métodos de melhoria.

	MMI	**MMS**	**MMT**
DPRM	9,47	9,31	9,30
Tempo (s)	1,46	1,76	3,37

3.3 Etapa 3: Construção de *Refset*

As melhores b_1 soluções do conjunto *P* são incluídas em *Refset*. Outras b_2 soluções são acrescentadas a *Refset* tal que a diversidade de *Refset* seja a maior possível. É definida uma distância entre duas soluções *x* e *y* para determinar a diversidade entre as mesmas: $d(x,y) = \sum_{k=1}^{k=n} h_k$, onde h_k é 0 se o cliente *k* pertence à mesma rota nas soluções *x* e *y*; é 1 caso contrário (as rotas são identificadas através de numeração das mesmas). Seleciona-se de *P* a solução que maximiza a distância mínima a todas as soluções já incluídas em *Refset*; o processo é repetido até serem inseridas deste modo b_2 soluções nesse conjunto.

3.4 Etapa 4: Geração de Subconjuntos

Terminada a construção de *Refset*, consideramos a implementação mais simples do *Método de geração de subconjuntos*, que consiste em realizar unicamente combinações de soluções duas a duas. Isto é, para a geração de cada subconjunto consideramos pares de soluções de *Refset* que não tenham sido ainda combinadas.

3.5 Etapa 5: Combinação de Soluções

Após selecionar os pares de soluções a serem combinadas aplica-se o *Método de Combinação de Soluções*. Este método funciona da seguinte maneira: i) Identifica-se os clientes que pertencem às mesmas rotas em soluções a serem combinadas. Esses clientes são

fixados às rotas; os clientes não fixados são desconectados das mesmas; ii) Constrói-se uma nova solução conforme explicado abaixo e iii) encontrada a nova solução, aplica-se à mesma o *Método de Melhoria* MMS.

Pode acontecer que após aplicado o método de combinação acima existam clientes desconectados das rotas, em cujo caso é necessária a viabilização da solução correspondente. Para isto é proposta uma heurística que considera os dados de distância e peso relativos do cliente que foi desconectado. Para cada cliente *i (cliente_candidato)* não fixado encontra-se a rota mais próxima. Uma rota se diz próxima do *cliente_candidato* se a distância do último cliente da mesma ao *cliente_candidato,* mais a distância do *cliente_candidato* ao depósito, tem o menor valor possível e satisfaz às restrições do PRV Clássico. A soma dessas duas distâncias é dividida pela demanda do *cliente_candidato.* O *cliente_candidato* com o menor valor deste quociente é inserido na rota mais próxima e os dados são atualizados. Quando a inserção de um novo cliente torna a solução inviável, uma nova rota é iniciada. O processo termina quando todos os clientes forem roteados.

Na Figura 6 ilustra-se o método de combinação de um problema com 15 clientes em que as duas soluções a serem combinadas têm três veículos cada. As Figuras 6(a) e 6(b) mostram as duas soluções que serão combinadas. Na Figura 6(c) clientes comuns das soluções *1* e *2* são identificados e fixados nas rotas respectivas; a obtenção da nova solução é feita de modo que sejam formadas rotas segundo o algoritmo descrito acima, conforme mostramos na Figura 6(d).

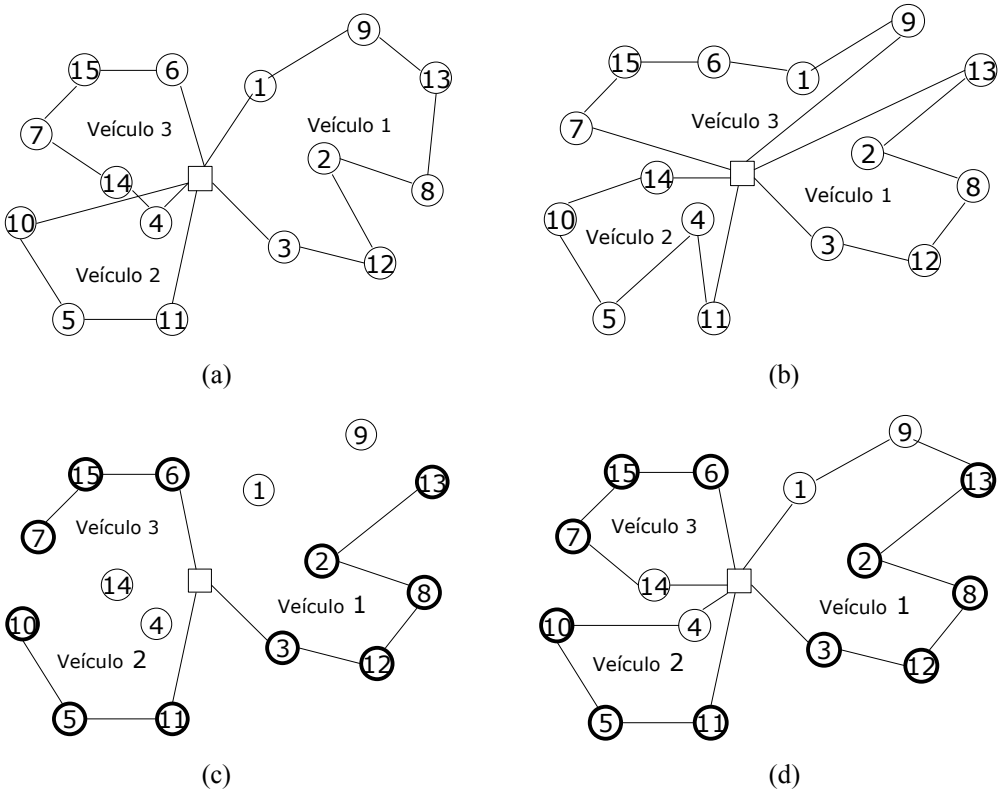


Figura 6 – (a) Solução 1; (b) Solução 2; (c) Clientes comuns; (d) Solução resultante.

3.6 Atualização do Conjunto de Referência

Se a solução encontrada pelo método de combinação for de qualidade, esta pode imediatamente substituir a pior solução em *Refset* (atualização dinâmica) ou ser armazenada em uma lista denominada *Pool* (para que futuramente seja efetuada uma atualização estática). No caso de uma atualização estática *Refset* consistirá das *b* melhores soluções armazenadas em $Refset \cup Pool$. A estratégia de atualização dinâmica modifica *Refset* rapidamente e usualmente produz soluções em menor tempo computacional, enquanto a estratégia de atualização estática usualmente produz melhores soluções, mas em mais tempo computacional.

A Tabela 3 mostra resultados obtidos com ambas as estratégias nas instâncias de Christofides *et alii* (1979) e Augerat *et alii* (1995) para diferentes valores de b_1 e b_2. Pode-se observar na Tabela que a atualização estática sempre tem um melhor desempenho em termos de desvio percentual relativo médio (DPRM), enquanto que o tempo computacional (Tempo) em que é encontrada a melhor solução durante o processo de busca é maior na atualização estática que na atualização dinâmica. Decidimos implementar o algoritmo de BD para o PRV Clássico utilizando a atualização estática.

Tabela 3 – Atualização Estática versus Atualização Dinâmica.

Tamanho *Refset* (BD_b_1_b_2)		**BD_5_5**	**BD_2_8**	**BD_3_7**	**BD_7_3**	**BD_10_10**
Atualização Estática						
Christofides	DPRM	0,86	0,98	1,09	1,08	0,78
	Tempo (s)	66,61	87,53	88,79	72,20	123,20
Augerat	DPRM	0,45	0,37	0,37	0,38	0,38
	Tempo (s)	12,93	2,86	2,84	3,09	1,22
Atualização Dinâmica						
Christofides	DPRM	2,15	1,92	1,46	2,33	1,44
	Tempo (s)	40,91	29,60	38,55	32,50	62,52
Augerat	DPRM	0,54	0,62	0,59	0,56	0,44
	Tempo (s)	2,61	2,50	2,40	2,25	1,11

Se na atualização de *Refset* novas soluções são introduzidas, ativa-se novamente o método da combinação de soluções. O critério de parada para a BD acontece quando foi atingido o valor ótimo da instância respectiva (quando disponível na literatura), ou quando *Refset* não foi modificado, isto é, quando não foi atualizado na última etapa da BD.

4. Resultados Computacionais

Para avaliar o desempenho do algoritmo de BD proposto na seção anterior efetuamos testes computacionais com quatro conjuntos de dados disponíveis na literatura, apresentados nas Tabelas A.1 a A.4 do Apêndice: o Conjunto I, formado pelas 14 instâncias descritas em Christofides *et alii* (1979); o Conjunto II, formado por 27 instâncias descritas em Augerat *et alii* (1995); o Conjunto III, formado por 11 instâncias descritas em Christofides & Eilon (1969) e o Conjunto IV, formado por 3 instâncias descritas em Fisher (1994). Nas instâncias do Conjunto I o número de clientes varia entre 50 e 199; no Conjunto II, entre 31 e 79; no Conjunto III, entre 21 e 100; e no Conjunto IV, entre 44 e 134 clientes.

Com a finalidade de permitir uma comparação apropriada, cabe esclarecer que a distância entre pares de vértices (clientes) é dada pela distância Euclidiana (sem nenhum arredondamento) para os Conjuntos I e IV, enquanto que para as instâncias dos Conjuntos II e III à distância entre pares de vértices é acrescentado o valor de 0.5, sendo então o valor obtido dessa soma arredondado. Além disso, conforme explicam Martinhon *et alii* (2004), o número de veículos para estas instâncias é um dado de entrada quando os problemas são resolvidos por meio de algoritmos exatos, mas pode ser uma variável de decisão no caso de heurísticas/ meta-heurísticas.

Os códigos para as diferentes implementações da BD foram programados na linguagem C, em um microcomputador PC com processador Pentium 4 de 2.26 GHz. Com o objetivo de avaliar o grau de proximidade entre as soluções encontradas nas diferentes implementações da BD em relação às melhores soluções encontradas na literatura, foi definida uma medida de *desvio*. Ela é dada pelo *desvio percentual relativo* (DPR), DPR = 100*((*BD*-*Fbest*)/*Fbest*), onde *BD* é o valor da função objetivo encontrado pela Busca Dispersa e *Fbest* é *a melhor solução disponível* na literatura para um dado problema-teste.

Na Tabela 4 mostramos os resultados computacionais obtidos pelo algoritmo de Busca Dispersa com diferentes valores de tamanho para *Refset*. A Tabela 4 mostra os resultados para diferentes estratégias na geração de *Refset*, utilizando dois tamanhos para o mesmo: 10 e 20 soluções. Várias combinações para ambos os tamanhos foram testadas. Os valores de b_1 e b_2 representam o número de soluções escolhidas por qualidade e diversidade, respectivamente. Para cada conjunto de instâncias a Tabela 4 mostra o desvio percentual relativo médio (DPRM) em relação às melhores soluções disponíveis na literatura (ver http://neo.lcc.uma.es/radi-aeb/WebVRP/); o tempo computacional (T) da BD em segundos; o tempo necessário para encontrar a melhor solução da BD (TE), em segundos; e o tempo computacional em segundos (TMD) para construir as soluções iniciais da BD. Todos os tempos mostrados na Tabela 4 são tempos médios, para o número de instâncias de cada conjunto.

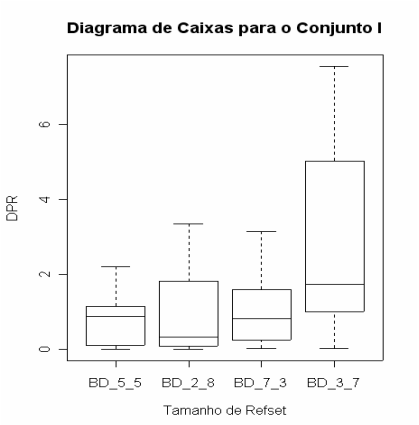
Tabela 4 – Resultados Computacionais da Busca Dispersa: Atualização Estática.

Instâncias		**BD_5_5**	**BD_2_8**	**BD_3_7**	**BD_7_3**	**BD_10_10**
Conjunto I	DPRM	0,86	0,98	1,09	1,08	0,78
(14 inst.)	T (s)	127,55	128	124,46	142,91	213,94
	TE(s)	66,61	87,53	88,79	72,20	123,20
	TMD(s)	33,32	32,35	35,30	34,46	47,83
Conjunto II	DPRM	0,45	0,37	0,37	0,38	0,38
(27 inst.)	T (s)	12,93	2,86	2,84	3,09	3,66
	TE(s)	1,96	0,70	0,48	0,60	1,22
	TMD(s)	1,66	0,87	0,87	0,86	0,83
Conjunto III	DPRM	0,14	0,08	0,03	0,05	0,12
(11 inst.)	T (s)	8,68	11,16	7,90	8,95	12,09
	TE(s)	2,88	2,97	1,94	2,59	4,34
	TMD(s)	3,91	5,38	3,85	3,84	4,19
Conjunto IV	DPRM	0,61	0,05	0,80	0,41	0,35
(3 inst.)	T (s)	32,27	21,91	26,76	32,01	42,75
	TE(s)	18,47	12,92	16,87	21,53	32,95
	TMD(s)	7,69	7,50	7,43	8,19	8,11

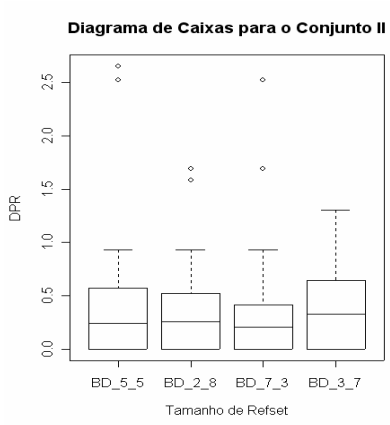
Foram realizados testes adicionais aumentando o tamanho de *Refset*, mas a melhoria dos resultados é pequena (quando o tamanho da instância é grande) ou não existe (instâncias pequenas), enquanto que o tempo computacional cresce bastante. As soluções obtidas pelo algoritmo da BD em todas as instâncias quando o tamanho de *Refset* é 10 e $b_1 = 5$ e $b_2 = 5$ são muito boas em termos de DPRM e tempo computacional, com um desvio percentual menor que 1% em relação às melhores soluções disponíveis na literatura. Em algumas instâncias foi possível igualar os melhores resultados disponíveis na literatura; em outras poucas instâncias (por exemplo na instância A-n37-k5.vrp de Augerat *et alii*, 1995) foram obtidas melhores soluções que as reportadas na literatura. É interessante notar que razoável percentagem do tempo computacional, na maioria das instâncias, é consumida ao gerar o conjunto de soluções iniciais (conjunto *P*).

Para uma melhor ilustração do comportamento do algoritmo da BD com os diferentes tamanhos de *Refset* apresentamos nas Figuras 7(a) a 7(d) o diagrama de caixas (ver Montgomery, 2004) para cada conjunto de dados. Os diagramas de caixas descrevem simultaneamente várias características importantes dos resultados obtidos, tais como a mediana (representada pela linha separadora no interior de cada caixa) e a dispersão dos DPR em relação ao DPRM (representada pela amplitude de cada caixa), entre outros. Os diagramas da Figura 7 foram obtidos rodando o *Software-R* (*The R Project for Statistical Computing*; para mais detalhes acessar http://www.r-project.org/).

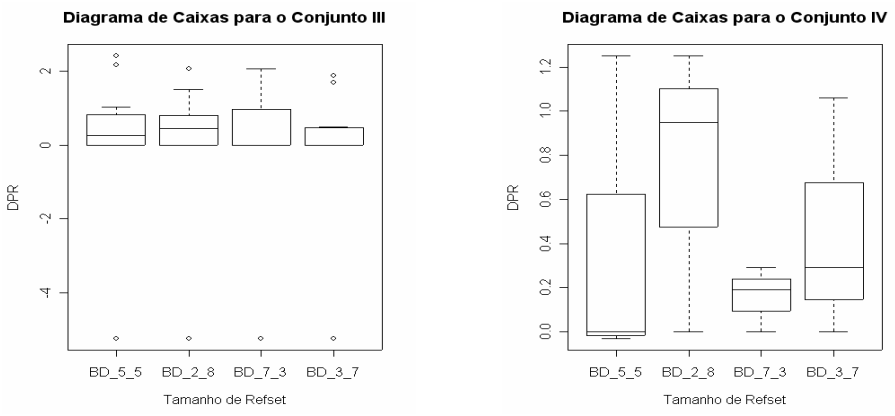
Na Figura 7(a) observamos que para os valores de $b_1 = 5$ e $b_2 = 5$ a maioria dos DPR's são inferiores à mediana, mostrando que o algoritmo de BD, para esses valores, tem um comportamento estável para o Conjunto I. Já quando $b_1 = 3$ e $b_2 = 7$ os resultados relativos aos dados do Conjunto I apresentaram a maior dispersão. A Figura 7(b) mostra que o algoritmo de BD teve o melhor desempenho para Conjunto II quando $b_1 = 7$ e $b_2 = 3$. Na Figura 7(c) o algoritmo da BD, para todos os diferentes valores de b_1 e b_2, tem um *outlier* (DPR = -5.24), pois em uma instância do Conjunto III foi encontrado um valor da função objetivo menor que o melhor resultado reportado na literatura; observa-se também que quando $b_1 = 3$ e $b_2 = 7$ o algoritmo tem seu melhor comportamento para os dados desse conjunto. Na Figura 7(d) apresenta-se o diagrama de caixas para o Conjunto IV, formado por apenas 3 instâncias.



(a)



(b)



(c) (d)

Figura 7 – Diagramas de Caixas para os Conjuntos Testados.

Se avaliarmos os diagramas da Figura 7 como um todo podemos afirmar que a composição de *Refset* para $b_1 = 5$ e $b_2 = 5$ tem em média um desempenho estável para todos os conjuntos de dados avaliados.

5. Conclusões e Perspectivas

Neste artigo descrevemos um algoritmo de Busca Dispersa desenvolvido para o PRV Clássico, testando seu desempenho para quatro conjuntos de dados disponíveis na literatura. Resultados computacionais mostram que as soluções encontradas pelo algoritmo de BD são bastante próximas às melhores soluções reportadas na literatura para essas instâncias, tendo sido obtidas em tempo computacional reduzido. Resultados ainda melhores poderão ser obtidos com a implementação de recursos avançados da BD, tais como por exemplo sua combinação com outras meta-heurísticas utilizadas com sucesso (e.g., Busca Tabu), para resolver tanto o PRV Clássico como PRV's com restrições adicionais.

Agradecimentos

Os autores agradecem ao CNPq pelo apoio financeiro recebido, na forma de uma bolsa de doutorado com duração de 4 anos para a autora e de uma bolsa de Pesquisador para o primeiro co-autor. Agradecem também os comentários e sugestões de avaliadores anônimos, que proporcionaram a melhoria do texto.

Referências Bibliográficas

(1) Augerat, J.R.; Belenguer, J.M. & Benevant, E. (1995). Computational results with a branch-and-cut code for the capacitated vehicle routing problem. *Technical Report RR940-M*, University Joseph Fourier, Grenoble.

(2) Baker, M.B. & Ayechew, M.A. (2003). A genetic algorithm for the vehicle routing problem. *Computers and Operations Research*, **30**, 787-800.

(3) Ball, M.O.; Magnanti, T.L.; Monma, C.L. & Nemhauser, G.L. [Editors] (1995). *Network Routing*. Handbook in Operations Research and Management Science, North Holland.

(4) Beasley, J.E. (1983). Route-first cluster-second methods for vehicle routing. *Omega*, **11**, 403-408.

(5) Campos, V.; Glover, F.; Laguna, M. & Martí, R. (2001). An experimental evaluation of a scatter search for the linear ordering problem. *Journal of Global Optimization*, **21**, 397-414.

(6) Christofides, N. & Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, **20**, 309-318.

(7) Christofides, N.; Mingozzi, A. & Toth, P. (1979). The vehicle routing problem. **In:** *Combinatorial Optimization* [edited by N. Christofides, A. Mingozzi, P. Toth, and C. Sandi], Wiley, Chichester, UK], 315-338.

(8) Cordeau, J-F.; Gendreau, M.; Laporte, G. & Semet, F. (2002). A guide to vehicle routing problems. *Journal of the Operational Research Society*, **53**, 512-522.

(9) Fisher, M.L. (1994). Optimal solution of vehicle routing problems using minimum *K*-trees. *Operations Research*, **42**, 626-642.

(10) Gillett, B. & Miller, L.R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, **22**, 340-349.

(11) Glover, F. (1977). Heuristic for integer programming using surrogate constraints. *Decision Sciences*, **8**, 156-166.

(12) Glover, F. (1997). A template for scatter search and path relinking. **In**: *Lecture Notes in Computer Science* [edited by J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers], Springer-Verlag, 13-54.

(13) Laguna, M. & Martí, R. (2000). Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Technical Report TR11-2000*, Departamento de Estadística e I.O., Universidad de Valencia.

(14) Laguna, M. & Martí, R. (2003). *Scatter Search Methodology and Implementations in C*. Kluwer Academic Publishers.

(15) Laporte, G.; Gendreau, M.; Potvin, J-Y. & Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, **7**, 285-300.

(16) Lenstra, J. & Rinnooy, K.A. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, **11**, 221-227.

(17) Lin, S. (1965). Computer solution of the traveling salesman problem. *Bell System Technical Journal*, **44**, 2245-2269.

(18) Martí, R.; Laguna, M. & Glover, F. (2006). Principles of scatter search. *European Journal of Operational Research*, **169**, 359-372.

(19) Martinhon, C.; Lucena, A. & Maculan, N. (2004). Stronger K-tree relaxations for the vehicle routing problem. *European Journal of Operational Research*, **158**, 56-71.

(20) Montané, F.A.T. & Galvão, R.D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research*, **33**, 595-619.

(21) Montgomery, D.C.; Runger, G.C. & Hubele, N.F. (2004). *Estatística Aplicada à Engenharia*. LTC – Livros Técnicos e Científicos Editora S.A., 2.ed., Rio de Janeiro, Brasil.

(22) Russell, R.A. & Chiang, W. (2006). Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research*, **169**, 606-622.

(23) Toth, P. & Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, U.S.A.

(24) Vigo, D. & Toth, P. (2003). The granular tabu search and its applications to the vehicle routing problem. *INFORMS Journal on Computing*, **15**, 335-346.

(25) Wassan, N.A. (2006). A reactive tabu search for the vehicle routing problem. *Journal of the Operational Research Society*, **57**, 111-116.

APÊNDICE

Tabela A.1 – Características das instâncias do Conjunto I.

Grupo	Instância	Nº. clientes	Capacidade do veículo	Tempo de serviço	Tempo máximo de rota	Melhor solução disponível na literatura
Dados distribuídos aleatoriamente						
	1	50	160	0	∞	524,61
	2	75	140	0	∞	835,26
I	3	100	200	0	∞	826,14
	4	150	200	0	∞	1028,42
	5	199	200	0	∞	1291,45
	6	50	160	10	200	555,43
	7	75	140	10	160	909,68
II	8	100	200	10	230	865,94
	9	150	200	10	200	1162,55
	10	199	200	10	200	1395,85
Dados distribuídos em agrupamentos						
III	11	120	200	0	∞	1042,11
	12	100	200	0	∞	819,56
IV	13	120	200	50	720	1541,14
	14	100	200	90	1040	866,37

Tabela A.2 – Características das instâncias do Conjunto II.

Instâncias	Nº. clientes	Veículos	Capacidade do veículo	Tempo de serviço	Tempo máximo de rota	Ótimo
A-n32-k5.vrp	31	5	100	0	∞	784
A-n33-k5.vrp	32	5	100	0	∞	661
A-n33-k6.vrp	32	6	100	0	∞	742
A-n34-k5.vrp	33	5	100	0	∞	778
A-n36-k5.vrp	35	5	100	0	∞	799
A-n37-k5.vrp	36	5	100	0	∞	669
A-n37-k6.vrp	36	6	100	0	∞	949
A-n38-k5.vrp	37	5	100	0	∞	730
A-n39-k5.vrp	38	5	100	0	∞	822
A-n39-k6.vrp	38	6	100	0	∞	831
A-n44-k6.vrp	43	6	100	0	∞	937
A-n45-k6.vrp	44	6	100	0	∞	944
A-n45-k7.vrp	44	7	100	0	∞	1146
A-n46-k7.vrp	45	7	100	0	∞	914
A-n48-k7.vrp	47	7	100	0	∞	1073
A-n53-k7.vrp	52	7	100	0	∞	1010
A-n54-k7.vrp	53	7	100	0	∞	1167
A-n55-k9.vrp	54	9	100	0	∞	1073
A-n60-k9.vrp	59	9	100	0	∞	1354
A-n61-k9.vrp	60	9	100	0	∞	1034
A-n62-k8.vrp	61	8	100	0	∞	1288
A-n63-k10.vrp	62	10	100	0	∞	1616
A-n63-k9.vrp	62	9	100	0	∞	1314
A-n64-k9.vrp	63	9	100	0	∞	1401
A-n65-k9.vrp	64	9	100	0	∞	1174
A-n69-k9.vrp	68	9	100	0	∞	1159
A-n80-k10.vrp	79	10	100	0	∞	1763

Tabela A.3 – Características das instâncias do Conjunto III.

Instância	Nº. clientes	Veículos	Capacidade do veículo	Tempo de serviço	Tempo máximo de rota	Ótimo
E-n22-k4.vrp	21	4	6000	0	∞	375
E-n23-k3.vrp	22	3	4500	0	∞	569
E-n30-k3.vrp	29	**3**	4500	0	∞	534
E-n33-k4.vrp	32	4	8000	0	∞	835
E-n51-k5.vrp	50	5	160	0	∞	521
E-n76-k7.vrp	75	7	220	0	∞	682
E-n76-k8.vrp	75	8	180	0	∞	735
E-n76-k10.vrp	75	10	140	0	∞	830
E-n76-k14.vrp	75	14	100	0	∞	1021
E-n101-k8.vrp	100	8	200	0	∞	815
E-n101-k4.vrp	100	14	112	0	∞	1067

Tabela A.4. Características das instâncias do Conjunto IV.

Instância	Nº. clientes	Veículos	Capacidade do veículo	Tempo de serviço	Tempo máximo de rota	Ótimo
F-n45-k4.vrp	44	4	2010	0	∞	723,54
F-n72-k4.vrp	71	4	30000	0	∞	241,97
F-n135-k7.vrp	134	7	2210	0	∞	1163,60