UNIVERSIDADE FEDERAL DE JUIZ DE FORA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

GUSTAVO CUNHA DE BITTENCOURT

MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA COMPUTACIONAL PARA A SOLUÇÃO DE UM PROBLEMA DE ROTEAMENTO DE VEÍCULOS (PRV) COM O USO DA METAHEURÍSTICA BUSCA DISPERSA (SCATTER SEARCH)

JUIZ DE FORA
2010

GUSTAVO CUNHA DE BITTENCOURT

MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA COMPUTACIONAL PARA A SOLUÇÃO DE UM PROBLEMA DE ROTEAMENTO DE VEÍCULOS (PRV) COM O USO DA METAHEURÍSTICA BUSCA DISPERSA (SCATTER SEARCH)

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do título de Engenheiro de Produção.

Orientador: DSc, Fernando Marques de Almeida Nogueira

JUIZ DE FORA

2010

Bittencourt, Gustavo Cunha de.

Modelagem e implementação de um sistema computacional para a solução de um problema de roteamento de veículos (PRV) com o uso da metaheurística busca dispersa (Scatter search) / Gustavo Cunha de Bittencourt. – 2010. -- 50 f..: il.

Trabalho de conclusão de curso (Graduação em Engenharia de Produção)-Universidade Federal de Juiz de Fora, Juiz de Fora, 2010.

1. Engenharia de produção. 2. Problema de roteamento de veículos. 3. Metaheurísticas. I. Título.

CDU 658.5

GUSTAVO CUNHA DE BITTENCOURT

MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA COMPUTACIONAL PARA A SOLUÇÃO DE UM PROBLEMA DE ROTEAMENTO DE VEÍCULOS (PRV) COM O USO DA METAHEURÍSTICA BUSCA DISPERSA (SCATTER SEARCH)

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do título de Engenheiro de Produção.

Aprovada em 11 de novembro de 2010.

BANCA EXAMINADORA

_____

DSc, Fernando Marques de Almeida Nogueira (Orientador)

Universidade Federal de Juiz de Fora

_____

MSc, Roberto Malheiros Moreira Filho

Universidade Federal de Juiz de Fora

_____

DSc, Marcos Martins Borges

Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Agradeço a Deus por mais esta vitória, pelo conhecimento que adquiri e pelo crescimento que obtive ao fim desta jornada. Agradeço também aos meus pais, Zilda e Bentos pelo amor, carinho e apoio constante, pelas palavras de força, conforto e incentivo, e por representarem meus exemplos de caráter e modelo de vida. Agradeço aos meus irmãos Rafael, Thiago, Ana, Anny e Alejandra por fazerem parte da minha vida e da minha família, sendo os alicerces de tudo o que é mais importante para mim. Agradeço também à minha madrinha Beth pela dedicação de quem trata um afilhado como um filho de coração, à minha avó Mimi e a todos os meus tios, primos, sobrinhos e amigos que me acompanharam nesta caminhada.

RESUMO

Dentre as atividades permeiam a cadeia de valor das organizações, a logística tem demonstrado especial importância. O impacto financeiro e estratégico que exerce leva um número cada vez maior de empresas a buscar soluções que permitam melhorias nos seus resultados, e como os transportes compõem grande parte do custo logístico, os sistemas de roteamento de veículos ganharam enorme destaque ultimamente. Este trabalho envolve a modelagem de um problema de roteamento de veículos (PRV), ou *vehicle routing problem* (VRP), e a implementação de um sistema computacional para resolvê-lo com o uso da metaheurística Busca Dispersa (*Scatter Search*). Um PRV consiste em atender a um determinado número de clientes, sem passar mais de uma vez por cada um deles, a partir de veículos que saem de um ou mais depósitos, aos quais devem retornar no final do trajeto. O seu objetivo é minimizar o custo total, distância total ou o tempo de percurso. Como os PRVs são de difícil solução à medida que o número de nós cresce, é inviável encontrar uma solução ótima pelos métodos convencionais de otimização. Neste caso, utilizam-se métodos heurísticos, ou mais recentemente metaheurísticas, que são procedimentos cujo objetivo é encontrar uma boa solução viável para resolver o problema em questão, mas não necessariamente a solução ótima.

Palavras-chave: Problema de roteamento de veículos. Metaheurísticas. Busca Dispersa.

ABSTRACT

Among the activities that permeate organizations' value chain, logistics has presented particular importance. Its financial and strategic impact have led a growing number of companies to seek ways to further improvements in their results, yet as transportation consists in a big part of the cost of supply chain, Vehicle Routing Systems have gained enormous prominence recently. This monograph involves the modeling of a vehicle routing problem (VRP), and the implementation of a computational system to solve that using the Scatter Search Metaheuristic. A VRP consists in visiting a certain number of customers by vehicles, which leave one or more deposits, which should be revisited at the end of the path without going more than once to the same client. Its objective is to minimize the total cost, total distance or travel time. VRPs are difficult to be solved as the number of nodes grows. So, it becomes infeasible to find an optimal solution by conventional methods of optimization. In this case, heuristics methods are used, or more recently metaheuristics ones, which are procedures whose goal is to find a good feasible solution in order to solve the problem, but not necessarily the optimal solution.

Keywords: Vehicle routing problem. Metaheuristics. Scatter Search.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

PRV	Problema de Roteamento de Veículos
PRVC	Problema de Roteamento de Veículos Capacitado
PRVD	Problema de Roteamento de Veículos com Restrição de Distância
PRVJT	Problema de Roteamento de Veículos com Janela de Tempo
PRPV	Problema de Roteamento Periódico de Veículos
BT	Busca Tabu
GRASP	Greedy Randomized Adaptive Search Procedure
GRASPf	Greedy Randomized Adaptive Search Procedure com filtro
SA	Simulated Annealing
BD	Busca Dispersa
SIG	Sistema de Informações Geográficas

SUMÁRIO

1. INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Dentre as atividades que envolvem a cadeia de valor das organizações, a logística tem mostrado uma especial importância, dado o impacto financeiro e estratégico que exerce sobre estas. Do ponto de vista estratégico, a gestão eficiente dos processos logísticos (e mais amplamente, da cadeia de suprimentos) pode se apresentar como vantagem competitiva, segundo Christopher (1997), sendo a fonte desta vantagem a diferenciação da organização aos olhos dos clientes e a redução dos custos de operação.

Vitasek (2010) define a logística como sendo o processo de planejamento, implementação e controle de procedimentos para o eficiente e eficaz transporte e armazenagem de bens, serviços e informações desde o ponto de origem até ao ponto de consumo com o objetivo de atender aos requisitos do cliente.

Com a busca constante por melhorias no atendimento e aumento da eficiência (incluindo reduções de custos e de tempo de entrega), as organizações investem cada vez mais em sistemas informatizados. Nos últimos seis anos, os operadores logísticos atuantes no Brasil procuraram ampliar a utilização de sistemas ERP e roteirizadores (INSTITUTO DE LOGÍSTICA E SUPPLY CHAIN, 2010), que tem por objetivo solucionar os problemas de roteamento de veículos (PRV).

O PRV foi proposto primeiramente por Dantzig e Ramster em 1959, e consiste em alocar uma frota veicular para o atendimento de uma determinada demanda de consumidores distribuídos geográfica ou espacialmente (OLIVEIRA, 2007).

1.2 JUSTIFICATIVA

A primeira justificativa para a realização deste trabalho é a sua relevância acadêmica, pois configura uma oportunidade notável de congregar os conhecimentos adquiridos na área de pesquisa operacional àqueles oriundos da área de logística.

Na maior parte das indústrias, a atividade de transporte representa um dos elementos mais importantes na composição do custo logístico. Nas nações desenvolvidas, os fretes costumam absorver até 60% do gasto logístico total (RODRIGUES, 2007). O modal rodoviário é responsável por 60% do total de cargas movimentadas no Brasil (chegando a 90%

no estado de São Paulo), e a frota de caminhões é da ordem de 1.800.000 veículos (TIGERLOG, 2010). Pesquisas indicam que serviços com transporte representam 2,53% do PIB brasileiro (FRAGA, 2007), demonstrando a representatividade deste setor para a economia brasileira, e a crescente procura por soluções para a redução destes custos. Nesta busca, sistemas que permitam maior eficiência no uso dos recursos, com a diminuição das distâncias percorridas nas rotas, melhorias no nível de atendimento e melhor alocação dos veículos disponíveis são demandados pelo mercado, que exige profissionais capacitados para o seu projeto, implantação, manutenção e operação.

Toth e Vigo (2002) atribuem grandes economias ao uso de procedimentos computadorizados para o planejamento da distribuição, variando de 5% a 20% do custo total de transportes, sendo estes custos responsáveis por até 20% do preço dos bens de consumo.

Outra justificativa para o uso de sistemas de roteamento de veículos é a redução de poluição, ruído e consumo de recursos naturais que estas soluções podem trazer à medida que diminuem o percurso efetuado por estes veículos, aumentando também a vida útil dos mesmos.

1.3 ESCOPO DO TRABALHO

O escopo deste trabalho engloba a modelagem matemática de um problema de roteamento de veículos capacitado (PRVC) e a implementação de um sistema computacional para resolvê-lo com o uso da metaheurística Busca Dispersa. Este modelo apresenta como restrição o somatório das cargas entregues por cada veículo, que não pode ultrapassar a sua capacidade, e tem como premissas a homogeneidade da frota. O sistema não suporta a utilização de janelas de tempo, entregas fracionadas e nem roteamento periódico de veículos (PRPV). Também não prevê a integração com SIGs (Sistemas de Informações Geográficas), embora estes sejam citados na revisão bibliográfica.

1.4 FORMULAÇÃO DE HIPÓTESES

Tem-se como hipótese deste trabalho que os resultados obtidos através de testes do sistema com o uso de instâncias da literatura serão próximos às melhores soluções relatadas pelos respectivos autores.

1.5 ELABORAÇÃO DOS OBJETIVOS

O objetivo geral deste trabalho é modelar matematicamente e computacionalmente um sistema de roteamento de veículos.

Objetivos específicos:

1. Solucionar o modelo com a utilização da metaheurística Busca Dispersa;
2. Efetuar testes de desempenho do sistema com instâncias da literatura.

1.6 DEFINIÇÃO DA METODOLOGIA

Dada a natureza aplicada da pesquisa realizada, o trabalho foi desenvolvido com base em uma adaptação das fases abordadas por Hillier e Lieberman (2006) para a elaboração de um estudo de pesquisa operacional:

1. Definição do problema de interesse;
2. Formulação de um modelo matemático para representar o problema;
3. Desenvolvimento de um procedimento computacional a fim de derivar soluções para o problema a partir do modelo;
4. Teste do modelo e aprimoramento conforme necessário.

Em seguida, o modelo foi alimentado com instâncias obtidas na literatura, a fim de comparar o seu desempenho com os resultados obtidos previamente pelos autores.

O estudo é explicativo e seguiu uma abordagem quantitativa, utilizando-se do método de pesquisa de modelagem e simulação. Neste método, um modelo matemático é utilizado como auxílio na tomada de decisões e resolução de problemas. Para tal, será utilizado o *software* de programação MathWorks MatLab® 7 R14 como ferramenta de desenvolvimento e teste do sistema.

1.7 ESTRUTURA DO TRABALHO

O primeiro capítulo do trabalho contém a parte introdutória, abordando as considerações iniciais, justificativas da escolha do tema, o escopo do trabalho, a formulação de hipóteses, elaboração dos objetivos, definição da metodologia e apresentação do cronograma.

O segundo capítulo contém a revisão bibliográfica, sendo esta desdobrada em uma visão geral acerca do problema de roteamento de veículos, uma abordagem dos métodos exatos para solução deste problema e uma exposição de algumas aplicações de métodos heurísticos.

O capítulo três contempla o desenvolvimento do trabalho, seguindo a metodologia citada no capítulo um. Na descrição do protocolo de pesquisa o problema abordado será definido e o modelo matemático demonstrado. Em seguida, será descrito o desenvolvimento do procedimento e relatados os aprimoramentos efetuados. Na descrição das unidades de análises serão explicitadas as instâncias de teste, retiradas da literatura.

O capítulo quatro apresenta os resultados obtidos, e estes são discutidos na seqüência.

Têm-se as conclusões do trabalho no capítulo 5, seguidas pelas referências bibliográficas.

2. REVISÃO DE LITERATURA

2.1 VISÃO GERAL

Um dos problemas mais famosos de otimização combinatória é o chamado Problema do Caixeiro Viajante, que consiste em determinar o circuito mais curto para se percorrer um dado número de pontos (chamados de nós) e retornar à origem, passando apenas uma vez por cada um deles (HILLIER e LIEBERMAN, 2006). Entretanto, este problema não reflete a realidade da maior parte das organizações, já que estas contam com uma série de veículos, que experimentam diversas restrições (de tempo e capacidade, por exemplo) e percorrem rotas distintas. Logo, o desafio destas empresas é determinar a melhor alocação dos veículos disponíveis, resolvendo um problema de roteamento de veículos (PRV).

O Problema de Roteamento de Veículos (PRV) é descrito como o problema de planejar a entrega ou coleção de rotas ótima. Estas rotas são compostas por veículos que devem partir de um ou vários depósitos para um determinado número de cidades ou clientes espalhados geograficamente, sujeito a um conjunto de restrições.

Laporte (1992) define o Problema Clássico de Roteamento de Veículos e mostra uma visão geral das diversas abordagens utilizadas para solucioná-lo. Estas se desdobram em algoritmos exatos, que encontram a solução ótima para o problema, e algoritmos heurísticos, que buscam uma boa solução viável, mas que não é necessariamente a solução ótima.

O PRV pode ser definido da seguinte forma: Seja $G = (V, A)$ um grafo onde $V = \{0, \ldots, n\}$ é um conjunto de vértices representando localidades (clientes ou cidades) com o depósito localizado no vértice 0, e A é o conjunto de arcos. Cada arco (i, j), $i \neq j$, é associado a uma matriz de distâncias $C = (c_{ij})$ não negativas. Em alguns contextos, c_{ij} também pode ser interpretado como o custo de viagem ou o tempo de viagem. Quando C é simétrico (isto é, a distância/tempo/custo de i para j é o mesmo de j para i), é conveniente substituir A por um conjunto E de arcos não direcionados. Além disso, assumimos que existem K veículos disponíveis no depósito, onde $K_L \leq K \leq K_U$. Quando $K_L = K_U$, K é dito ser fixo. Quando $K_L = 1$ e $K_U = n - 1$, K é dito ser livre. Quando K não é fixo, faz sentido associar um custo fixo f ao uso do veículo. Como simplificação, Laporte (1992) ignorou estes custos, e partiu-se do princípio de que todos os veículos são idênticos e têm a mesma capacidade Q. O PRV consiste em planejar um conjunto de rotas de menor custo do veículo, de tal forma que:

(i) Cada vértice em $V\backslash\{0\}$ é visitado apenas uma vez e por exatamente um veículo;

(ii) Todas as rotas se iniciam e terminam no depósito;

(iii) As seguintes restrições devem ser respeitadas:

a. Restrição de capacidade: a cada vértice $i > 0$ é atribuído um peso não-negativo (ou demanda) d_i e a soma dos pesos de qualquer rota do veículo não pode exceder a capacidade do veículo;

b. O número de vértices em cada rota é limitado a q (este é um caso especial de (a) com $d_i = 1$ para todo $i > 0$ e $Q = q$);

c. Restrição de tempo total: o comprimento de qualquer rota não pode exceder um limite fixado L, sendo este comprimento constituído pelos tempos de viagem c_{ij} e pelos tempos de parada s_i em cada vértice i da rota;

d. Janelas de tempo: o vértice i deve ser visitado dentro do intervalo de tempo $[a_i, b_i]$ e é permitido tempo de espera no vértice i;

e. Precedência entre pares de vértices: o vértice i pode ter de ser visitado antes do vértice j.

Esta lista não é exaustiva, e uma série de outras variantes interessantes são descritas na literatura.

PRVs de capacidade limitada são designados como PRVCs, PRVs com restrição de tempo ou distância máxima da rota são designados PRVDs, e PRVs com janelas de tempo são designados PRVJTs. Além destes, são comuns as variantes PRVFH, onde a frota de veículos é heterogênea, e PRVEF, onde as entregas são fracionadas.

2.2 ALGORITMOS EXATOS

Os algoritmos exatos para o PRV são classificados por Laporte (1992) em três grandes categorias: métodos de busca direta em árvore (são utilizados como exemplos a atribuição de limite inferior e um algoritmo *branch-and-bound*, e a árvore geradora central de grau k), programação dinâmica (tendo como exemplo uma de suas formulações) e programação linear inteira (utilizando como exemplos o particionamento de conjuntos e um algoritmo de geração de colunas, a formulação de um índice triplo de fluxo de veículos e a formulação de um índice duplo de fluxo de veículos).

A atribuição de limite inferior e um algoritmo *branch-and-bound* relacionado explora a relação entre o PRV e uma de suas relaxações, o *m*-PCV (Problema do Caixeiro Viajante). Dado o grafo $G = (V, A)$ com um depósito no vértice 0 e os m veículos com base no depósito, o *m*-PCV consiste em estabelecer m rotas de custo mínimo começando e terminando no depósito, e de tal forma que cada vértice restante seja visitado exatamente uma vez (LAPORTE, 1992). Para tal, é estabelecido um limite superior m_u para m, e o *m*-PCV pode ser transformado em um 1-PCV, seguindo os seguintes passos:

Etapa 1. Aumentar o número de vértices através da introdução de $m_u - 1$ depósitos artificiais; sendo $n' = m_u + n - 1, V' = (1, \ldots, n')$ e $A' = A \cup \{(i, j) : i, j \in V', i \neq j, i \, ou \, j \in V' \backslash V\}$.

Etapa 2. Gerar uma matriz de distâncias estendida $C' = (c'_{ij})$ associada a A' é definida por:

$$c'_{ij} = \begin{cases} c_{ij} & (i, j \in V), \\ c_{i1} & (i \in V \backslash \{0\}, j \in V' \backslash V), \\ c_{1j} & (i \in V' \backslash V, j \in V\{0\}), \\ \gamma & (i, j \in (V' \backslash V) \cup \{1\}, \end{cases} \quad (1)$$

onde o valor de γ depende da variante do problema considerada:

$\gamma = \infty$ resulta na distância mínima para m_u veículos;

$\gamma = 0$ resulta na distância mínima para até m_u veículos;

$\gamma = -\infty$ resulta na distância mínima para o número mínimo de veículos;

Sendo x_{ij} $(i \neq j)$ uma variável binária igual a 1 se e somente se o arco (i, j) aparecer na solução ótima, o PRV pode ser formulado da seguinte maneira:

Minimizar $\sum_{i \neq j} c'_{ij} x_{ij}$ (2)

Sujeito a $\sum_{j=0}^{n'} x_{ij} = 1 \qquad (i = 0, \dots, n')$ (3)

$\sum_{i=0}^{n'} x_{ij} = 1 \qquad (j = 0, \dots, n')$ (4)

$\sum_{i,j \in S} x_{ij} \leq |S| - v(S) \qquad (S \subset V'\{0\};\ |S| \geq 2)$ (5)

$x_{ij} \in \{0,1\} \qquad (i, j = 0, \dots, n';\ i \neq j)$ (6)

A função objetivo e as restrições (3), (4) e (6) desta formulação compõem um problema de designação modificado (são proibidas designações na diagonal principal). A restrição (5) elimina subcircuitos, onde $v(S)$ é um limite inferior do número de veículos necessários para visitar todos os vértices de S na solução ótima (LAPORTE, 1992). Esta restrição é obtida levando em conta que para cada $S \subset V'\{0\}, |S| = 2,\ \bar{S} = V' \backslash V$, teremos:

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq v(S) \tag{7}$$

e que a seguinte identidade é válida:

$$|S| = \sum_{i,j \in S} x_{ij} \sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \,. \tag{8}$$

O valor de $v(S)$ depende do tipo de PRV considerado. No caso de PRVCs, é válido considerar:

$$v(S) = \left\lceil \frac{\sum_{i \in S} d_i}{Q} \right\rceil. \tag{9}$$

No PRVD, $v(S)$ não é tão fácil de determinar. Entretanto, um limite inferior para seu valor geralmente pode ser determinado durante o *branch-and-bound*. Caso contrário, é sempre válido assumir $v(S) = 1$.

É interessante observar que a restrição (5) desempenha um papel duplo: garante que todas as rotas de veículos satisfaçam às restrições de capacidade máxima ou de comprimento máximo de rota, como também garante que a solução não contenha subcircuitos desconectados do depósito, uma vez que cada subconjunto S de $V\backslash\{0\}$ será ligado ao seu complemento (LAPORTE, 1992).

O problema é então resolvido através de um processo de *branch-and-bound* onde os subproblemas são problemas de designação. A única diferença reside na definição de subcircuitos ilegais. Estes passaram a incluir:

(i) subcircuitos sobre um conjunto z de vértices $V\backslash\{0\}$;

(ii) as rotas dos veículos que violam as restrições de capacidade máxima ou de comprimento máximo: trata-se de caminhos de vértices $(i_1, i_2, \dots, i_r)$, onde $i_1,\ i_r \in \{1, n+1, \dots, n+\ m_u - 1\},\ i_2, \dots, i_{r-1} \in V\{0\}$ e $\sum_{t=1}^{r-1} d_{i_t} > Q$ ou $\sum_{t=0}^{r-1} c_{i_t, i_{t+1}} > L$.

Estes podem ser eliminados através do particionamento do subproblema inviável atual, considerando $v(S) = 1$.

Para interpretar o resultado como uma solução do PRVC, aplicamos as seguintes regras: considere um arco (i, j), pertencente à solução (LAPORTE, 1992):

(i) se $i \in V\backslash\{0\}$ e $j \in V'\backslash V$, substitua (i, j) por $(i, 0)$,

(ii) se $j \in V\backslash\{0\}$ e $i \in V'\backslash V$, substitua (i, j) por $(0, j)$,

(iii) se $i, j \in V'\backslash V$, apague (i, j).

Na literatura constam PRVCs assimétricos gerados aleatoriamente de até 260 vértices otimizados com esta metodologia.

A técnica da árvore geradora central de grau k e um algoritmo relacionado foi desenvolvida por Christofides, Mingozzi e Toth para PRVs simétricos, definidos pelo grafo $G = (V, E)$, onde $c_{ij} = c_{ji}$. Ele é baseado na relaxação da árvore geradora central de grau k do $m-$PCV, onde m é fixo. Em qualquer solução possível, o conjunto E de arcos pode ser dividido em quatro subgrupos, sendo:

E_0: arcos que não pertençam à solução;

E_1: arcos formando uma árvore geradora central de grau k, ou seja, uma árvore geradora sobre G, onde o grau do vértice *0* é igual a k (com $k = 2m - y$);

E_2: y arcos E_2 que incidem sobre o vértice *0* ($0 \leq y \leq m$);

E_3: $y - m$ arcos que não incidem sobre o vértice *0*. Em seguida, são inseridas variáveis binárias (ver LAPORTE, 1992) e uma das restrições é relaxada através da relaxação Lagrangeana (ver ESPEJO e GALVAO, 2002). Laporte (1992) incorporou o limite inferior do número de rotas constituídas pelo depósito e um único vértice em $V\backslash\{0\}$, definido na etapa anterior, em um algoritmo *branch-and-bound*, e foram resolvidos com sucesso PRVs que continham de 10 a 25 vértices.

A programação dinâmica foi proposta para PRVs por Eilon, Watson-Gandy e Christofides, e pode ser formulada da seguinte forma, considerando um número m fixo de veículos: seja $c(S)$ o custo (ou comprimento) de uma rota do veículo através do vértice *0* (depósito) e de todos os vértices de um subconjunto S de $V\backslash\{0\}$, e $f_k(U)$ o custo mínimo atingível usando k veículos e entregando para um subconjunto U de $V\backslash\{0\}$. Então, o custo mínimo pode ser determinado através da seguinte expressão (LAPORTE, 1992):

$$f_k(U) = \begin{cases} c(U) & (k = 1) \\ \min\limits_{U^* \subset U \subseteq V\backslash\{0\}} [\, f_{k-1}(U\backslash U^*) + c(U^*)] & (k > 1) \end{cases} \qquad (10)$$

O custo da solução é igual a $f_m(V\backslash\{0\})$ e a solução ótima corresponde à otimização dos subconjuntos U^* na equação acima. Se $f_k(U)$ tiver de ser calculado para todos os valores de k e para todos os subconjuntos U de $V\backslash\{0\}$, o número de cálculos exigido será excessivo na maior parte dos problemas. Para que a programação dinâmica seja eficiente, faz-se necessária uma redução substancial do número de estados por meio de um processo de relaxação, ou usando os critérios de viabilidade e dominância. Segundo Laporte (1992), com o uso de relaxações (como a relaxação de espaço de estados, por exemplo), limites inferiores em soluções ótimas foram obtidas em 10 problemas de roteamento que continham de 10 a 25 vértices. A razão "limite inferior/solução ótima" variou entre 93,1% e 100%. Christofides (1985b, apud LAPORTE, 1992) relatou que PRVCs com até 50 vértices podem ser resolvidos de forma sistemática com esta abordagem.

Segundo Laporte (1992), o particionamento de conjunto e geração de colunas (*set partitioning and column generation*) teve em Balinski e Quandt uma de suas primeiras proposições para PRVs. Considere J o conjunto de todas as rotas possíveis j e a_{ij} um coeficiente binário igual a *1* se e somente se o vértice $i > 0$ aparecer na rota j. Seja c_j^* o custo ótimo da rota j e x_j, uma variável binária igual a *1* se e somente se a rota j é usada na solução ótima. O problema pode ser formulado como:

$$\text{Minimizar} \quad \sum_{j \in J} c_j^* x_j \tag{11}$$

$$\text{Sujeito a} \quad \sum_{j \in J} a_{ij} x_j = 1 \qquad (i \in V \backslash \{0\}) \tag{12}$$

$$x_j \in \{0,1\} \qquad (j \in J) \tag{13}$$

Laporte (1992) listou as duas dificuldades principais associadas a esta formulação:

(i) Em casos reais, o número de variáveis binárias x_j a serem executadas pode chegar próximo aos milhões;

(ii) O cálculo dos valores de c_j^* apresenta grande dificuldade.

Uma maneira de contornar estes empecilhos é a utilização de um algoritmo gerador de colunas. Neste algoritmo, um problema reduzido contendo apenas um subconjunto de todas as possíveis colunas (variáveis) é resolvido repetidamente. A relaxação linear do problema reduzido fornece um vetor dual λ ótimo. A verificação de otimalidade implica no cálculo do menor custo marginal da coluna s. Como as soluções do PRV devem ser inteiras, este procedimento precisa ser utilizado conjuntamente ao algoritmo *branch-and-bound*. Na literatura (ver LAPORTE, 1992) foram relatadas soluções para PRVs com janela de tempo de até 100 vértices utilizando esta metodologia.

A formulação de um índice triplo de fluxo de veículos foi desenvolvida por Fisher e Jaikumar em 1978 para solucionar PRVs com restrições de capacidade e janela de tempo, mas sem considerar os tempos de parada s_i. Esta formulação utiliza variáveis x_{ijk} para representar a passagem de um veículo por um arco, onde x_{ijk} recebe o valor 1 se e somente se o veículo k passa pelo arco (i, j) na solução ótima, e 0 se ele não passa $(i \neq j)$. Laporte (1992) explica que embora este algoritmo aparentemente tenha sido utilizado apenas para fornecer uma solução heurística para o problema, ele garante uma solução ótima em um número finito de etapas, caso seja executado até a conclusão. Este problema pode ser subdividido em outros dois: o problema de alocação generalizada (PAG), obtido pela relaxação de algumas variáveis; e um PCV (Problema do Caixeiro Viajante) com janelas de tempo (PCVJT). A solução proposta por Fisher e Jaikumar foi baseada na decomposição de Benders (ver LAPORTE, 1992), e estes reportaram resultados para PRVs de até 199 vértices.

No caso de PRVs simétricos, uma formulação mais simples pode ser obtida retirando-se o índice *k* das variáveis, resultando na formulação do índice duplo de fluxo de veículos. Neste caso, x_{ij} $(i < j)$ indica quantos arcos (i, j) são atravessados por um veículo, sendo que

$x_{ij} \in \{0,1\}$ caso i e j sejam clientes $(i > 0)$, ou $x_{ij} \in \{0,1,2\}$ caso i seja o depósito $(i = 0)$. Desta forma, o problema pode ser resolvido com uma extensão do algoritmo para o PCV desenvolvido por Laporte (1992), tendo sido utilizado para PRVs de até 60 vértices.

2.3 HEURÍSTICAS E METAHEURÍSTICAS

Uma heurística pode ser definida como um algoritmo que encontra uma boa solução viável para um determinado problema, em um tempo razoável, mas sem a garantia de que esta é a melhor solução. Segundo Hillier e Lieberman (2006), o procedimento normalmente é um algoritmo iterativo completo em que cada iteração envolve a condução de uma busca por uma nova solução que, eventualmente, poderia superar o melhor resultado encontrado previamente. Quanto o algoritmo termina após um tempo razoável, a solução por ele fornecida é a melhor que foi encontrada durante qualquer iteração.

As metaheurísticas são uma classe de heurísticas mais recentes, que possuem como diferencial uma série de ferramentas que reduzem o risco de paradas prematuras em ótimos locais ainda distantes de um ótimo global. Geralmente estas ferramentas são componentes aleatórios inseridos durante a execução do algoritmo, que permitem que outras zonas de soluções sejam exploradas.

O PRV pertence à categoria de problemas NP – difíceis (OLIVEIRA, 2007), cuja dificuldade de encontrar a solução aumenta rapidamente à medida que o número de nós cresce. Laporte (1992) cita e exemplifica uma série de algoritmos exatos, mas reconhece a limitação que estes apresentam ao lidar com problemas maiores e mais complexos, indicando as metaheurísticas como um campo de estudos mais promissor.

As heurísticas aplicadas à solução de PRVs podem ser divididas, segundo Toth e Vigo (2002), em três categorias: métodos construtivos, como o algoritmo das economias de Clarke e Wright; métodos de duas fases, que se subdividem em *route-first, cluster second* e *cluster-first, route-second*; e métodos de melhoria, que se subdividem em melhoria intra-rotas e melhoria inter-rotas.

A heurística de Clarke e Wright é um exemplo de método construtivo. Ela foi proposta em 1964 para resolver PRVCs cujo número de veículos é livre. Segundo Laporte (1992), o método possui três passos básicos, começando com n rotas contendo o depósito $(i = 0)$ e outro vértice i, no formato $(0, i, 0)$. Em cada etapa, duas rotas são mescladas de acordo com a maior economia que pode ser gerada, sendo duas rotas $(0, \ldots, i, 0)$ e $(0, j, \ldots, 0)$ fundidas em

uma única rota $(0, \ldots, i, j, \ldots, 0)$ e a economia computada pela fusão das duas rotas dada por $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. O algoritmo é executado em um tempo $O(n^2 \log n)$, mas pode ser simplificado com a utilização de estruturas de dados adequadas.

Nos métodos de duas fases, têm-se a heurística de Beasley como um exemplo de *route-first, cluster second.* Segundo Sosa, Galvão e Gandelman (2007), resolve-se um problema do caixeiro viajante (com algoritmos exatos ou heurísticos) partindo-se depósito e regressando a este no fim. Em seguida formam-se os clusters, tomando como base as restrições do problema de roteamento considerado (capacidade do caminhão, tempo de percurso, etc), e gerando-se assim uma solução viável.

O algoritmo de varredura (*sweep algorithm*) é atribuído a Gillet e Miller, embora segundo Toth e Vigo (2002) tenha sido originalmente publicado por Wren em 1971. Este algoritmo é um exemplo de método de duas fases *cluster-first*, *route-second*, e os vértices devem ser representados por suas coordenadas polares (θ_i, ρ_i), onde θ_i é o ângulo e ρ_i é o comprimento do raio. A um vértice i^* arbitrário deve ser atribuído o valor $\theta_i^* = 0$, e os ângulos restantes devem ser computados a partir do arco $(0, i^*)$, como se o giro de uma semi-reta varresse os demais clientes. Os vértices devem ser ordenados em ordem crescente de θ_i, e o seguinte procedimento deve ser seguido (LAPORTE, 1992):

Etapa 1. Escolha um veículo não utilizado k.

Etapa 2. A partir do vértice não roteado com o menor ângulo, atribuir vértices para o veículo k enquanto a sua capacidade não for excedida. Caso permaneçam vértices não roteados, retorne à Etapa 1.

Etapa 3. Resolver para cada rota um PCV (exato ou aproximado), visando encontrar a melhor distribuição inter-rota.

Etapa 4. Realize trocas de vértices entre rotas adjacentes e verifique se a distância total diminui. Re-otimize e pare.

Dos algoritmos de melhoria intra-rotas, os mais conhecidos são os movimentos λ-opt, nos quais segmentos com λ vértices (sendo λ um número inteiro) são retirados das rotas e a ordem dos elementos é alterada buscando a redução de distâncias. Num movimento 3-opt, por exemplo, se os vértices $[a, b, c]$ da rota forem retirados, será procurada entre as seguintes seqüências aquela que minimiza a distância percorrida pelo veículo naquela rota: $[a, b, c]$;$[a, c, b]$; $[b, a, c]$; $[b, c, a]$; $[c, a, b]$; e $[c, b, a]$. Os movimentos λ-opt são realizados em um tempo computacional $O(n^{\lambda})$ (TOTH e VIGO, 2002).

Dentre os algoritmos de melhorias inter-rotas, destacam-se os movimentos de re-alocação, intercâmbio e cruzamento. O movimento de re-alocação consiste na retirada de um cliente de uma rota e na tentativa de inseri-lo em cada uma das demais, sempre buscando o movimento que minimiza a distância total. O movimento de intercâmbio ocorre com a troca de clientes entre rotas, duas a duas; e o movimento de cruzamento consiste na seleção de um par de rotas e na divisão de cada uma delas em dois segmentos. O primeiro segmento da primeira rota selecionada é conectado ao segundo segmento da segunda rota selecionada, e vice-versa (SOSA, GALVÃO e GANDELMAN, 2007).

No campo das metaheurísticas, Laporte (1992) aponta a Busca Tabu (BT) como um dos métodos mais promissores na busca de boas soluções para os PRVs, dado o sucesso com que foi aplicada a diversos problemas desta natureza. Esta é uma metaheurística que constrói uma seqüência de soluções e, em seguida, executa um passo de melhoria. Como o algoritmo pode gerar rotas inviáveis, o grau de viabilidade de partida das rotas é medido por meio de penalidades na função objetivo.

Ochi e Tortelly (2003) apresentaram uma nova metaheurística híbrida baseada em conceitos de *Greedy Randomized Adaptive Search Procedure* (GRASP) e Busca Tabu (BT) para a solução do Problema de Roteamento Periódico de Veículos (PRPV). O PRPV é uma variação do PRV onde ocorre *scheduling*, isto é, os pedidos são programados para serem entregues em um período de T dias. Quando T=1, o PRPV se restringe ao clássico Problema de Roteamento de Veículos (PRV).

Cada cliente no PRPV deve ser visitado k vezes, onde $1 \leq k \leq T$ e no modelo clássico, a demanda diária de um cliente é sempre igual para cada dia de visita. O objetivo do PRPV pode ser visto como a de gerar um conjunto de rotas para cada dia de planejamento, de modo que as restrições envolvidas sejam atendidas e os custos globais minimizados.

Como o objetivo deste estudo não é modelar um PRPV (e sim um PRV), optou-se por dar maior ênfase ao algoritmo desenvolvido por Ochi e Tortelly (2003), em detrimento das especificidades do PRPV. A estrutura do algoritmo se baseia em um GRASP, onde a fase de busca local é feita através de um método de Busca Tabu.

O GRASP foi desenvolvido em 1995 por Feo e Resende, e consiste de um processo onde cada iteração possui duas fases: uma fase de construção, na qual uma solução viável é construída; e uma fase de melhoria, cujo objetivo é encontrar uma solução ótima local.

Segundo Ochi e Tortelly (2003), a fase de construção é iterativa, pois a solução inicial é construída elemento a elemento; é adaptativa, porque os benefícios associados a cada

elemento são levados de uma iteração para outra; e pode ser gulosa, porque a adição de cada elemento pode estar restrita a uma lista de apenas um candidato. Nesta fase, a escolha do próximo elemento a ser adicionado é determinada pela ordenação de todos os elementos candidatos em uma lista de candidatos C. Esta lista é construída de acordo com uma função gulosa $g : C \rightarrow \Re$, que mede o benefício de se selecionar cada elemento. O componente probabilístico do GRASP é caracterizado pela escolha aleatória de um dos melhores elementos da lista de candidatos, não sendo necessariamente o melhor.

A fase de melhorias consiste em um procedimento de busca local, já que a solução gerada na fase de construção do GRASP pode não ser um ótimo global. Um algoritmo de busca local consiste na substituição da solução corrente pela melhor solução da vizinhança desta. Na maioria dos casos a busca se encerra quando nenhuma solução melhor é encontrada na vizinhança (OCHI e TORTELLY, 2003).

Para montar cada solução inicial do PRPV, ou seja, para gerar um conjunto de rotas para cada dia do horizonte de planejamento, respeitando as restrições associadas, foi escolhida uma alternativa de visita para cada cliente aleatoriamente. A seguir, para cada dia do horizonte de planejamento, um método das pétalas foi executado através do algoritmo de varredura (*sweep algorithm*) de Gillet e Miller para gerar as rotas diárias.

Uma segunda versão do GRASP proposto por Ochi e Tortelly (2003) utiliza na fase de construção um filtro para selecionar as melhores soluções iniciais. Neste caso, a cada iteração do GRASP são geradas k soluções iniciais (onde k é um parâmetro de entrada) usando o algoritmo de varredura e somente a melhor solução do PRPV é selecionada para a fase de busca local do GRASP. Esta versão com filtro foi denominada GRASPf.

A cada iteração do GRASP, uma etapa de busca local é efetuada utilizando um método de Busca Tabu (BT). O método Busca Tabu proposto incorpora etapas de busca intensiva e etapas de diversificação utilizando memória de curto e longo prazo. A solução inicial (semente) é a solução gerada na fase de construção do GRASP, e a partir desta semente o método Busca Tabu passa a explorar diferentes regiões do conjunto de soluções.

Cada região está associada a uma busca local do método, onde uma procura intensiva é efetuada. A troca de regiões de busca representa uma etapa de diversificação. A busca local da Busca Tabu equivale a verificar a possibilidade de cada vértice de um PRV diário trocar de rota e/ou trocar de rota e dia (OCHI e TORTELLY, 2003).

Segundo Ochi e Tortelly (2003), no método Busca Tabu proposto por Cordeau, Gendreau e Laporte cada vértice do PRV diário tenta ser inserido (ou permutado) em cada

uma das demais rotas de todos os dias do horizonte de planejamento. Isso torna o processo de busca extremamente oneroso. Como alternativa Ochi e Tortelly (2003) propuseram para cada rota diária rd uma vizinhança V_{rd} desta rota, tal que os vértices somente são permutados (ou inseridos) com os demais vértices de V_{rd}, conforme determinado pelo ângulo α mostrado na ilustração 1, abaixo.

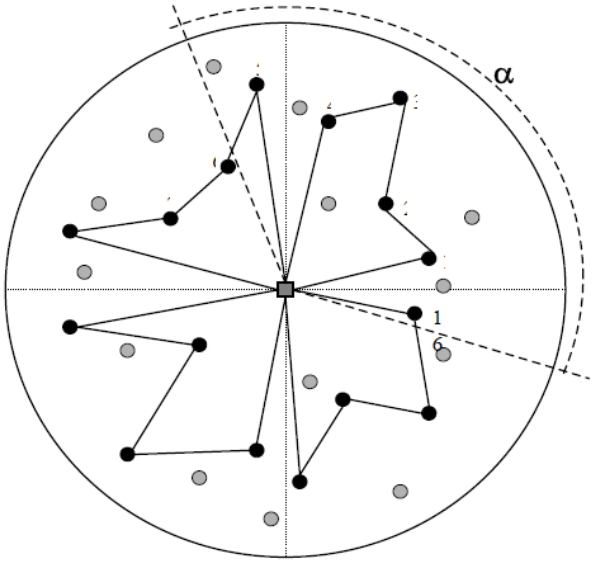

Ilustração 1 – Vizinhança da rota diária rd para permutação
Fonte: Ochi e Tortelly, 2003 (Extraído).

A melhor solução resultante desta busca local, será considerada a nova semente para inicializar uma nova busca local no método Busca Tabu. Este procedimento será repetido m vezes, onde m é um parâmetro de entrada.

Neste método, um movimento da solução semente a uma solução vizinha equivale a permutar componentes da lista tripla (i, r, d) [vértice (cliente) i, alocado na rota r, no dia d]. Quando uma alocação (i, r, d) é efetuada, este movimento é mantido tabu durante certo número de iterações, ou seja, o vértice i não poderá ser retirado da rota r e nem do dia d durante aquelas iterações (OCHI e TORTELLY, 2003).

A etapa de diversificação do método BT é ativada quando se atualiza uma iteração do algoritmo GRASP. Ou seja, após m buscas locais do método BT, uma nova semente é construída de forma independente através do método de construção do GRASP.

Testes foram realizados por Ochi e Tortelly (2003) utilizando instâncias listadas na literatura. Na comparação entre as duas versões do GRASP (com e sem filtro) e a versão da Busca Tabu desenvolvida no artigo, o GRASPf mostrou uma superioridade evidente na qualidade das soluções obtidas. Outro aspecto importante a se notar é que a introdução do

módulo de filtro no GRASPf não onera o tempo final quando comparado com a versão GRASP sem filtro. Segundo Ochi e Tortelly (2003), isso se explica pelo fato da geração de soluções não demandar tempos significativos quando comparados aos tempos gastos na busca local. O GRASPf também demonstrou desempenho superior quando comparado a outras heurísticas presentes na literatura, obtendo o melhor resultado em 7 das 10 instâncias nas quais foi submetido a testes.

Galvão et al. (1997) apresentam a integração de um sistema PRV (resolvido com a utilização da metaheurística *Simulated Annealing*) a um software SIG (Sistema de Informações Geográficas) para a visualização e manipulação das rotas geradas.

Segundo Galvão et al. (1997), *Simulated Annealing* (SA) é uma metaheurística sobreposta a métodos iterativos de busca local e representa uma variação de métodos tradicionais descendentes de otimização local ou busca por vizinhança. Seja um problema de otimização combinatória representado por um par (Ω, Z), onde Ω é o conjunto de todas as soluções viáveis e Z é uma função objetivo a ser minimizada ou maximizada. Em um problema de minimização busca-se uma solução ótima $S^* \in \Omega$ tal que $Z(S^*) \leq Z(S) \; \forall \; S \in \Omega$.

Um mecanismo de geração de vizinhanças define para cada solução $S \in \Omega$ o conjunto de todas as soluções $S' \in \Omega$ que podem ser geradas de acordo com certas regras. Uma transição de S para S' em $N(S)$ é chamada de movimento e $N(S)$ é chamada de vizinhança de S. Uma solução S é o mínimo local em relação ao mecanismo de geração se e somente se $Z(S) \leq Z(S') \; \forall \; S' \in \mathrm{N(S)}$ (GALVÃO et al., 1997).

Em algoritmos descendentes de busca local apenas movimentos que reduzam o valor da função objetivo são aceitos. Já o *Simulated Annealing* ocasionalmente aceita movimentos que aumentem o valor da função objetivo, por intermédio de uma estratégia de aceitação probabilística. O algoritmo aceita um movimento de S para $S' \in N(S)$ que piore a solução com probabilidade $e^{-\Delta/T}$, na qual $\Delta = Z(S') - Z(S)$ e T é um parâmetro positivo chamado temperatura. Os valores de T são atualizados segundo uma função de redução de temperatura, e o processamento termina quando um dos critérios de parada é atingido (GALVÃO et al., 1997).

A aplicação de *Simulated Annealing* a um problema de otimização combinatória exige a definição de (GALVÃO et al., 1997):

(i) uma solução inicial viável para o problema;

(ii) um mecanismo de geração de vizinhanças;

(iii) uma estratégia de seleção de soluções vizinhas;

(iv) valor inicial do parâmetro de controle (temperatura T_0);

(v) função de redução de temperatura;

(vii) critérios de parada.

No trabalho de Galvão et al. (1997) foi elaborado um PRVD (PRV com restrição de tempo ou distância), com o objetivo de minimizar a distância total viajada pela frota de veículos. Cabe lembrar que outras funções objetivo podem ser utilizadas em PRVs, como a minimização do tempo de percurso, do número de veículos alocados para suprir a demanda total, ou ainda a minimização do custo total de entrega.

Galvão et al. (1997) desenvolveram o algoritmo dentro de um programa denominado FazRota, cuja função é extrair as informações do roteamento (endereços de depósito e clientes, informações sobre a frota de entrega, demandas e tempos de serviço dos clientes, comprimentos máximos de rotas) de uma base de dados, e associar os vértices às interseções de ruas e clientes a serem visitados. Em seguida, é gerado um grafo direcionado, levando em consideração a direção do tráfego, e o algoritmo de Floyd é aplicado para a geração da matriz de distância entre os vértices, contendo as distâncias dos arcos (i,j) (sendo $i \neq j$). Estes dados alimentam o procedimento baseado em *Simulated Annealing*, que processa a solução e gera os arquivos de interface com o Sistema de Informações Geográficas MapInfo.

No software MapInfo, o usuário pode visualizar as rotas obtidas em mapas e realizar algum tratamento que se mostre necessário. O programa FazRota ainda gera um relatório detalhado das rotas estabelecidas como solução do problema, através de um roteiro seqüencial que orienta o motorista e os seus auxiliares durante as entregas.

Segundo Galvão et al. (1997) , o algoritmo *Simulated Annealing* utilizado no programa apresentou ótimo desempenho nos testes realizados com uso de instâncias presentes na literatura, superando alguns dos melhores resultados já mencionados em algumas delas. Entretanto, o tempo de processamento de algoritmos baseados em *Simulated Annealing* tende a ser relativamente longo, e segundo Toth e Vigo (2002) estes métodos não produzem resultados competitivos se comparados a outras metaheurísticas, como a Busca Tabu, por exemplo.

2.4 BUSCA DISPERSA (*SCATTER SEARCH*)

A Busca Dispersa (*Scatter Search*) tem sido utilizada por diversos autores para a solução de PRVs complexos, como relatado em Rego e Leão (2001); Sosa, Galvão e Gandelman (2007); e Belfiore et al. (2007), bem como foi a metaheurística utilizada para a solução do PRV proposto neste trabalho.

A Busca Dispersa é uma metodologia com base em populações, e tem demonstrado ser muito efetiva na solução de problemas de otimização discreta. Apesar de a Busca Dispersa apresentar algumas semelhanças com os Algoritmos Genéticos, ela se 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 (SOSA, GALVÃO e GANDELMAN, 2007).

Esta metaheurística 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, com base em estratégias para combinar regras de decisão e restrições.

A metodologia 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. Neste contexto, a classificação boa não se restringe apenas à qualidade da solução, mas também à sua diversidade em relação a outras soluções deste conjunto (SOSA, GALVÃO e GANDELMAN, 2007).

A Busca Dispersa é composta basicamente por cinco etapas (métodos):

1. *Etapa Geradora de Soluções Iniciais*: esta etapa gera um conjunto *P* (conjunto de soluções obtidas com o método gerador de soluções iniciais) com *Psize* (tamanho da população *P* de soluções iniciais) soluções, de onde é extraído 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, tanto através de métodos de melhoria intra-rotas quanto inter-rotas.

3. *Geração do Conjunto de Referência*: nesta etapa é construído ou atualizado o conjunto de referência. O *Refset* é composto por b soluções, sendo $b = b_1 + b_2$, onde b_1 é o

número de soluções de qualidade que compõem o conjunto, isto é, as soluções com menor distância total, e b_2 é o número de soluções de diversidade que compõem o *Refset*, isto é, as soluções que mais se distinguem das demais que já o constituem. Esta distinção é calculada através da máxima diferença mínima entre uma determinada rota e todas as demais do *Refset*, sendo esta diferença entre duas rotas definida como o somatório de um fator que recebe 1 caso o vértice i não pertença à mesma rota em duas soluções, e 0 caso ele pertença.

3.1 Construção – A partir do conjunto P se extrai *Refset*, composto pelas b_1 melhores soluções (em ordem crescente de distância total) e pelas b_2 soluções mais diversas.

3.2 Atualização – A atualização do *Refset* acontece quando novas soluções que atendem aos requisitos para ingressar no conjunto são encontradas. Assim, o *Refset* mantém seu tamanho constante, mas as soluções pertencentes ao mesmo melhoram durante o processo de busca.

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.

Testes de desempenho realizados com instâncias da literatura revelaram um ótimo desempenho do algoritmo BD implementado por Sosa, Galvão e Gandelman (2007), mostrando desvios percentuais muito baixos em relação aos melhores resultados obtidos na literatura, e em algumas instâncias até superando estes resultados.

3. DESENVOLVIMENTO

3.1 DESCRIÇÃO DO PROTOCOLO DE PESQUISA

O desenvolvimento do trabalho seguiu conforme descrito na seção 1.6 – Definição da Metodologia, com base em uma adaptação das fases abordadas por Hillier e Lieberman (2006) para a elaboração de um estudo de pesquisa operacional.

3.1.1 DEFINIÇÃO DO PROBLEMA ABORDADO

Neste trabalho foi modelado um PRV capacitado e com frota homogênea. Este pode ser definido da seguinte forma: Seja $G = (V, A)$ um grafo onde $V = \{0, \ldots, n\}$ é um conjunto de vértices representando localidades (clientes ou cidades) com o depósito no vértice 0, e A é o conjunto de arcos. Cada arco (i, j), $i \neq j$, é associado a uma matriz de distâncias $C = (c_{ij})$ não negativas. Embora os dados utilizados neste trabalho configurem o problema abordado como sendo um PRV simétrico (isto é, $c_{ij} = c_{ji} \; \forall \, i, j \in V, i \neq j$), optou-se por não substituir o conjunto A de arcos direcionados por um conjunto E de arcos não direcionados dada a possibilidade de perfeita utilização deste modelo também com dados assimétricos. Além disso, assume-se que existem K veículos disponíveis no depósito, sendo K livre, isto é, $K_L \leq K \leq K_U$, $K_L = 1$ e $K_U = n - 1$, mas nenhum custo fixo f foi associado ao uso dos veículos, já que este dado não estava disponível nas instâncias de teste utilizadas. Parte-se também do princípio de que todos os veículos são idênticos e têm a mesma capacidade Q. As seguintes premissas e restrições devem ser respeitadas:

1. Cada vértice em $V \backslash \{0\}$ é visitado apenas uma vez e por exatamente um veículo;
2. Todas as rotas se iniciam e terminam no depósito;
3. Restrição de capacidade: a cada vértice $i > 0$ é atribuído um peso não-negativo (ou demanda) d_i e a soma dos pesos de qualquer rota do veículo não pode exceder a capacidade Q do veículo.

3.1.2 MODELO MATEMÁTICO DO PROBLEMA ABORDADO

O PRV Capacitado abordado neste trabalho pode ser formulado seguindo o modelo matemático de índice triplo de fluxo de veículos adotado por Toth e Vigo (2002), conforme abaixo:

$$\text{Minimizar} \quad \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij} \sum_{k=1}^{K} x_{ijk} \tag{14}$$

$$\text{Sujeito a} \quad \sum_{k=1}^{K} y_{ik} = 1 \qquad (i = i, \dots, n) \tag{15}$$

$$\sum_{k=1}^{K} y_{0k} = K \tag{16}$$

$$\sum_{j=0}^{n} x_{ijk} = \sum_{j=0}^{n} x_{jik} = y_{ik} \qquad (i = 0, \dots, n), (k = 1, \dots, K) \tag{17}$$

$$\sum_{i=1}^{n} d_i y_{ik} \leq Q \qquad (k = 1, \dots, K) \tag{18}$$

$$\sum_{i \in S}\sum_{j \in S} x_{ijk} \leq |S| - 1 \qquad (\forall\, S \subseteq V\backslash\{0\}\,, |S| \geq 2, k = 1, \dots, K) \tag{19}$$

$$x_{ijk} \in \{0,1\} \qquad (i, j = 0, \dots, n\,;\, i \neq j)\,, (k = 1, \dots, K) \tag{20}$$

$$y_{ik} \in \{0,1\} \qquad (i, j = 0, \dots, n\,;\, i \neq j), (k = 1, \dots, K) \tag{21}$$

A função objetivo (14) busca a minimização da distância total percorrida, onde a variável binária x_{ijk} recebe 1 se o arco (i, j) é coberto pelo veículo k na solução ótima, e 0 se não. A variável binária y_{ik} recebe o valor 1 caso o vértice i seja visitado veículo k na solução ótima, e 0 caso não seja. Portanto, a restrição (15) desta formulação impõe que cada vértice i seja visitado por apenas um veículo, a restrição (16) determina que K veículos devem deixar o depósito e a restrição (17) estabelece que o mesmo veículo k deve entrar e sair de um determinado vértice i. A restrição de capacidade (18) determina que a carga de qualquer veículo k não pode ultrapassar a sua capacidade Q. A restrição (19) elimina subcircuitos ilegais, determinando que para qualquer subconjunto S de $V\backslash\{0\}$ o número total de arcos x_{ijk} $(j, i \in S)$ pertencentes à solução ótima deve menor ou igual ao número de elementos de S menos um, conforme pode ser visto na ilustração 2. As últimas restrições, (20) e (21), indicam que as variáveis x_{ij} e y_{ik} são binárias, isto é, podem receber apenas os valores 1 e 0.

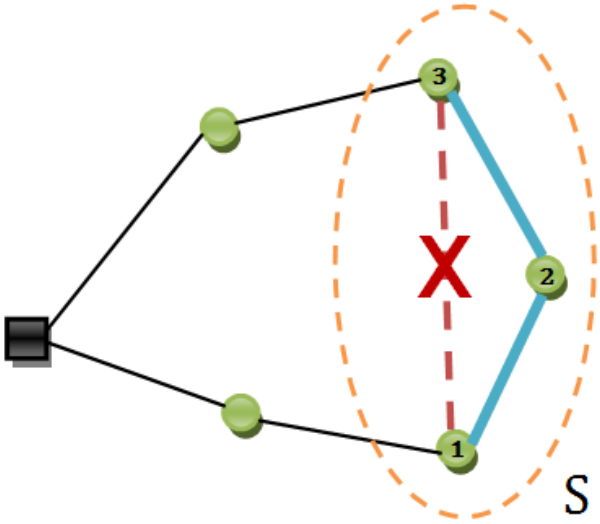

Ilustração 2 – Eliminação de subtours ilegais
Fonte: Organizado pelo autor.

3.1.3 DESCRIÇÃO DO DESENVOLVIMENTO

O Problema de Roteamento de Veículos Capacitado definido e matematicamente modelado nas etapas anteriores foi resolvido com a utilização da metaheurística Busca Dispersa.

Por pertencer à categoria de problemas NP – difíceis, a busca por soluções ótimas para PRVs com algoritmos exatos se mostra extremamente limitada e em alguns casos até inviável, conforme citado por Laporte (1992). Portanto, optou-se por excluir a utilização de tais métodos do escopo do trabalho. O uso isolado das heurísticas descritas também apresenta grandes deficiências, em especial no que tange a qualidade das soluções encontradas, inviabilizando o seu emprego nesta forma. Entretanto, quando combinadas e utilizadas como componentes das metaheurísticas, se revelam como fator decisivo na busca de boas soluções em tempos viáveis. Dentre as metaheurísticas abordadas na revisão bibliográfica, o *Simulated Annealing* foi apontado como não sendo competitivo para a solução de PRVs, restando o GRASP, a Busca Dispersa e a Busca Tabu. Destes, a Busca Dispersa foi escolhida dado o enorme destaque que tem recebido ultimamente por parte de diversos autores e pelos resultados que tem alcançado na solução de inúmeros problemas de roteamento de veículos.

O sistema foi implementado utilizando a linguagem computacional própria do *software* MathWorks MatLab® 7 R14, com um script principal conectado a uma série de módulos para o processamento de funções específicas. Um detalhe a ser notado nesta

linguagem computacional é que a declaração prévia de variáveis no início do script não se faz necessária.

O módulo principal se inicia com a definição dos valores das variáveis b_1 e b_2, explicadas anteriormente. Em seguida o sistema abre uma janela na qual o usuário pode selecionar o conjunto de dados que deseja utilizar, devendo este estar no formato de texto (extensões .txt ou .dat), organizado matricialmente. No primeiro valor da primeira linha deve constar o número n de clientes; separado por um espaço simples, o segundo valor deve conter a capacidade Q do veículo; e o terceiro valor da primeira linha deve apresentar o tempo máximo de rota D permitido para cada veículo (sendo que este tempo máximo não será considerado pelo sistema modelado, já que este trabalha exclusivamente com as restrições de capacidade do veículo). O primeiro valor da segunda linha deve conter o tempo de serviço s, que também não será considerado por este modelo pelos mesmos motivos citados anteriormente. O segundo e o terceiro valores da segunda linha do arquivo de dados contêm, respectivamente, a coordenada x e a coordenada y do depósito. Da terceira linha em diante, o primeiro valor contém sempre a coordenada x do vértice representado por aquela linha, o segundo valor contém a coordenada y e o terceiro valor contém a demanda d_i associada àquele vértice.

Os dados coletados são então inseridos em uma estrutura de registro denominada *vértices*, que contém os seguintes campos: *id*, que é o código de identificação de cada vértice, iniciando-se por 0 no depósito; *linha*, que determina a posição (número da linha) que cada vértice ocupa no registro, iniciando-se também pelo depósito (linha 1); *x*, que contém a coordenada x do vértice armazenado naquela linha; *y*, que contém a coordenada y do vértice armazenado naquela linha; *demanda*, que contém a demanda do cliente armazenado naquela linha (no caso do depósito – linha 1 – a demanda é zero); *x0*, que contém a coordenada x do vértice armazenado naquela linha em relação ao depósito, sendo o depósito considerado a origem do sistema de coordenadas $(0,0)$; *y0*, que contém a coordenada y do vértice armazenado naquela linha em relação ao depósito (sendo que para o depósito o *y0* também é zero); e *theta*, que contém o ângulo da coordenada polar daquele vértice num sistema de coordenadas onde a origem é o depósito. O *theta* de um determinado vértice é calculado a partir do arco cuja tangente é dada pela razão $y0/x0$, convertido para valores positivos em função do quadrante em que este vértice se encontra.

O passo seguinte do algoritmo é criar uma matriz *distancias*, que contém a distância euclidiana calculada entre todos os pontos pertencentes à estrutura de registro *vértices*. É

válido lembrar que o cálculo desta matriz ocorre desta forma porque os conjuntos de dados utilizados para teste do sistema eram simétricos e as demais referências bibliográficas que também utilizaram estas instâncias assim o fizeram. Entretanto, a matriz *distancias* pode ser um dado de entrada do problema, obtida através da interface com SIGs, por exemplo, já que o modelo é perfeitamente capaz de trabalhar com matrizes de custos (ou distâncias) assimétricas.

Com os dados inseridos no sistema, inicia-se a etapa geradora de soluções iniciais. A heurística utilizada para a geração de tais soluções foi o algoritmo de varredura (*sweep algorithm*) de Gillet e Miller. Um vetor de permutação aleatória é preenchido com n (número de clientes) elementos, e destes são utilizados os primeiros *PSize* vértices como referência para o início da rota. Para cada vértice i sorteado, é construída uma matriz *SementeTheta* n x 2 que contém na primeira coluna o id do vértice e na segunda o seu *theta*, ambos coletados do registro *vertices*. É assumido que o *theta* do vértice i é igual a zero, e os valores de *theta* dos demais vértices são recalculados em função deste primeiro, tendo como referência um sistema cartesiano onde o eixo x é representado pela reta que passa pelo depósito e pelo vértice i e o eixo y perpendicular a esta reta, passando pelo depósito, sendo considerado positivo o sentido anti-horário. Com a matriz preenchida, os demais vértices são ordenados pelo seu *theta* recalculado, de maneira crescente, sendo utilizada como critério de desempate a menor distância entre o vértice e o depósito (ρ), no caso de dois vértices possuírem o mesmo *theta*.

Uma matriz tridimensional *soluções* é criada, contendo três índices (v, k, s). O índice s indica qual a solução que está selecionada, isto é, qual das matrizes bidimensionais de índices (v, k) está sendo utilizada. O k é o índice das colunas da matriz bidimensionais, e indica uma determinada rota (veículo) da solução s; e o índice v indica a posição daquele determinado vértice dentro da rota k. A primeira rota (coluna $k = 1$) da primeira solução da matriz *soluções* ($s = 1$) é então preenchida com os valores de *id* dos vértices ordenados na primeira matriz *SementeTheta* criada enquanto o somatório das demandas dos vértices alocadas a esta rota respeitar a restrição de capacidade Q. Quando esta restrição for desrespeitada, inicia-se uma nova rota ($k = 2$) na mesma solução ($s = 1$), e este processo é repetido até que todos os vértices ordenados em *SementeTheta* sejam alocados às suas respectivas rotas. A distância total percorrida pelos veículos na solução recém criada é calculada por um módulo que utiliza como base a matriz *distâncias*, e os procedimentos de melhoria são executados.

O primeiro procedimento de melhoria realizado é a melhoria intra-rota, com a utilização da heurística 2-opt. Para cada solução criada, esta heurística realiza a troca entre

todos os vértices dentro de cada uma das rotas, dois a dois ($\lambda = 2$), sendo que a cada troca a solução resultante é armazenada em uma matriz auxiliar. A distância total percorrida pelos veículos nesta matriz é calculada e, caso esta seja inferior à distância calculada previamente, a solução s é substituída pela matriz auxiliar. Em seguida, a solução é submetida seqüencialmente aos três procedimentos de melhoria inter-rotas: Re-alocação, intercâmbio e cruzamento.

A heurística de re-alocação remove cada vértice v de cada rota $k = w$ (onde w é o índice de um laço que percorre todas as rotas de uma dada solução s) e tenta inseri-lo em cada posição v de cada um das demais rotas $k \neq w$ de uma matriz auxiliar criada como cópia da solução s. A distância total percorrida pelos veículos nesta matriz é calculada e, caso esta seja inferior à distância calculada previamente, a solução s é substituída pela matriz auxiliar.

Em seguida executa-se a heurística de intercâmbio que, para cada vértice v de cada rota $k = w$, tenta realizar a troca deste com cada vértice v de cada um das demais rotas $k \neq w$ (de uma matriz auxiliar criada como cópia da solução s). Novamente a distância total é calculada para cada movimento de intercâmbio na matriz auxiliar e, caso este seja inferior à distância calculada previamente, a solução s é substituída por esta matriz.

O último movimento de melhoria inter-rotas executado é a heurística de cruzamento, que corta, para cada posição v de uma matriz auxiliar criada como cópia da solução s, duas rotas vizinhas e realiza a troca de seções. Esta troca ocorre com o encaixe da primeira parte da primeira rota com a segunda parte da segunda rota, e vice versa. Os cruzamentos são realizados entre cada rota $k = w$ e as demais rotas $k \neq w$, duas a duas, e armazenados na matriz auxiliar, que tem a distância total calculada e, caso esta seja inferior à distância calculada previamente para a solução s, esta é substituída pela nova solução.

Por fim, executa-se novamente heurística 2-opt, e com isto se tem uma das soluções iniciais criadas. Este processo de construção se repete, começando de cada um dos *PSize* vértices sorteados no vetor de permutação aleatória, até que se tenha *PSize* soluções iniciais.

Quando todas as soluções inicias estão armazenadas na matriz *soluções*, dá-se início ao processo de construção do *Refset*. Este processo é divido em duas etapas, sendo a primeira a seleção das b_1 soluções de menor distância total, e a segunda etapa dada pela seleção das b_2 soluções mais distantes, conforme definido na seção 2.4.

Um vetor é preenchido com a distância total de cada uma das *PSize* soluções iniciais da matriz *soluções*, e em seguida é classificado em ordem crescente. Os espaços do vetor *Refset* de 1 até b_1 são preenchidos pelos índices s das soluções ordenadas anteriormente.

Entretanto, como existe a possibilidade de que haja soluções idênticas apenas com o índice das rotas alterados (por exemplo, a primeira rota da solução s_1 ser igual à terceira rota da solução s_2, e assim para todas as demais rotas), foi criado um algoritmo auxiliar que avalia a equivalência entre as rotas, e só permite que uma nova solução seja inserida no *Refset* se ela for diferente de todas as outras já presentes, evitando a redundância de combinações e esforço computacional desnecessário na etapa seguinte.

Os b_2 espaços seguintes do *Refset* são preenchidos pelas soluções que apresentem maior divergência em relação àquelas já presentes no conjunto. É calculada a divergência entre duas soluções através do somatório de um fator que recebe 1 caso o vértice i não pertença à mesma rota nas duas soluções, e 0 caso ele pertença. Para cada solução não pertencente ao *Refset*, é calculada a sua divergência em relação a cada uma das soluções presentes no conjunto de referência, e a menor destas divergências é armazenada em um vetor auxiliar, que é organizado em ordem decrescente em seguida. Caso a solução correspondente ao primeiro valor deste vetor auxiliar seja diferente de todas as demais soluções do *Refset* (neste caso, com o sentido de não ter rotas equivalentes às de nenhuma das soluções já contidas no conjunto), esta solução é adicionada ao conjunto de referência. O processo se repete até que as b_2 soluções mais divergentes sejam armazenadas no *Refset*.

Com o *Refset* criado, são gerados os subconjuntos de soluções que serão combinadas. Estes subconjuntos são gerados dois a dois entre todas as soluções presentes no *Refset*, e armazenados em uma matriz de duas colunas chamada *subconjuntos_combinações*, onde o elemento da primeira coluna representa a primeira solução a ser combinada e o elemento da segunda coluna representa a segunda solução. Em seguida é criada uma outra matriz de duas colunas chamada *combinações_feitas*, que é cópia da matriz *subconjuntos_combinações*.

Um módulo de combinação de soluções é executado, que percorre a matriz *subconjuntos_combinações* e combina a solução da primeira coluna com a solução da segunda coluna para cada linha percorrida da seguinte forma: As duas soluções são comparadas, e um algoritmo estabelece quais são as rotas equivalentes entre elas (são classificadas rotas equivalentes nesta situação como aquelas que possuem maior número de vértices em comum nas duas soluções, em um processo semelhante ao que ocorre no cálculo da divergência entre as soluções). Para cada rota da primeira solução, os vértices que não pertencem também à rota estabelecida como sua equivalente na segunda solução são retirados e armazenados em um vetor chamado *pool_vertices*, e os que pertencem a ambas as rotas são fixados em uma nova solução criada na matriz tridimensional *soluções*. Um módulo de

inserção é executado, inserindo os clientes armazenados no *pool_vertices* na rota mais próxima da nova solução, segundo um critério definido por Sosa, Galvão e Gandelman (2007): Para cada cliente i não fixado encontra-se a rota mais próxima, que é aquela onde a distância do último cliente alocado na rota ao cliente i, mais a distância do cliente i ao depósito é a menor possível e satisfaz às restrições do problema. O cliente a ser inserido na rota mais próxima é aquele que apresentar o menor valor de um fator dado pelo somatório destas duas distâncias dividido pela demanda do cliente. O cliente com menor valor deste fator é inserido na rota mais próxima da nova solução, e o processo de inserção se reinicia, até que todos os clientes do *pool_vertices* sejam alocados. A nova solução passa então pelos métodos de melhoria já citados anteriormente (2-opt, re-alocação, intercâmbio, cruzamento, 2-opt, na respectiva ordem), e o processo de combinação de soluções se repete até que todas as linhas da matriz *subconjuntos_combinações* sejam percorridas.

Ao fim do procedimento anterior, o *Refset* é atualizado novamente, mas desta vez com a nova matriz tridimensional *soluções*, e são gerados os subconjuntos de soluções que serão combinadas, mas com algumas diferenças. A primeira delas é que a matriz *subconjuntos_combinações* recebe apenas os subconjuntos que não foram anteriormente combinados, sendo esta precedência conferida através de uma verificação na matriz *combinações_feitas*. Ao fim da geração dos subconjuntos a serem combinados na próxima iteração, as novas combinações a serem efetuadas são adicionadas também à matriz *combinações_feitas*, e estes módulos são inseridos em um laço que executa estas operações enquanto a matriz *subconjuntos_combinações* não estiver vazia. Quando esta matriz estiver vazia é sinal de que o *Refset* não foi atualizado no último laço, tanto em relação às soluções de qualidade (b_1) quanto em relação às soluções de diversidade (b_2) e, portanto, o programa finaliza a sua execução.

Com a finalização de execução, os seguintes outputs são apresentados ao usuário: o número da melhor solução encontrada (que é o primeiro valor de b_1 do último *Refset* gerado), para que esta possa ser identificada dentro da matriz *soluções* (que contém tanto as soluções criadas inicialmente quanto aquelas resultantes do método de combinação); a menor distância encontrada, que é a distância total percorrida por todos os veículos na solução supracitada; o tempo de processamento gasto para a criação das soluções iniciais; o tempo de processamento total do aplicativo; e a plotagem de um gráfico que mostra as rotas geradas de maneira intuitiva e simples, conforme será apresentado na seção referente aos resultados.

3.1.4 TESTES E APRIMORAMENTOS

Diversos testes foram realizados paralelamente ao desenvolvimento do sistema com a finalidade de identificar erros encontrados durante a implementação do algoritmo. Os erros de sintaxe são de fácil correção e são exibidos na tela durante a execução do script e/ou função. Entretanto, a correção de erros de lógica de programação foi indubitavelmente a parte mais complexa do desenvolvimento do trabalho, dada a dificuldade de se identificar exatamente onde se encontravam com a utilização da ferramenta de *debug* do *software* MathWorks MatLab® 7 R14. Esta ferramenta exige que o algoritmo seja executado passo a passo, suspendendo a sua execução a cada iteração para a busca de falhas. Este movimento ganha enorme dificuldade quando o possível erro se encontra em um laço que possui um número considerável de iterações.

Dentre as falhas detectadas, a de mais difícil tratamento ocorreu no módulo de inserção dos vértices da etapa de geração de subconjuntos, que foi resolvido após uma série de testes com o uso da ferramenta de *debug*. Um erro de atualização das matrizes que fez com que em algumas soluções dois dos vértices do *pool_vertices* inseridos na mesma rota fossem sobrepostos na matriz auxiliar, e conseqüentemente na matriz *soluções*. Com isso, um dos clientes era excluído do roteamento, inviabilizando a busca pela melhor solução, os procedimentos de melhoria e as combinações desta solução.

Dentre as melhorias efetuadas ao longo do desenvolvimento do sistema, cabe destacar o algoritmo de verificação de igualdade aplicado às soluções a serem inseridas no *Refset*, que evita a redundância de combinações e esforço computacional desnecessário na etapa seguinte, e o algoritmo de verificação de equivalências entre as rotas na etapa de combinação, evitando que vértices fossem removidos desnecessariamente das suas rotas e armazenados no *pool_vertices*.

3.2 DESCRIÇÃO DAS UNIDADES DE ANÁLISE

Os testes computacionais do sistema, tanto para correção de falhas quanto para validação dos procedimentos adotados, foram realizados com a utilização das quatro primeiras instâncias de testes de Christofides, Mingozzi e Toth, conforme descrito em Sosa, Galvão e Gandelman (2007). O conjunto de dados possui quatorze instâncias no total, mas apenas as quatro primeiras não possuem limitação de tempo máximo da rota, o que permite que os resultados obtidos sejam comparados com os disponíveis na literatura. As instâncias

estão disponíveis para acesso público através do endereço http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html. Os detalhes das instâncias podem ser vistos na tabela 1, sendo que a última coluna representa a distância total da melhor solução disponível na literatura, segundo Sosa, Galvão e Gandelman (2007):

Tabela 1 – Detalhes das instâncias de Christofides

Instância	Nº de Clientes	Capacidade do Veículo	Melhor Solução Disponível
1	50	160	524,61
2	75	140	835,26
3	100	200	826,14
4	150	200	1028,42

Fonte: Sosa, Galvão e Gandelman (2007)

4. RESULTADOS

4.1 RESULTADOS ALCANÇADOS

Como resultado do projeto, obteve-se um sistema de roteamento de veículos perfeitamente funcional e estável após vinte e três revisões do código principal, além das versões desenvolvidas para cada um dos algoritmos auxiliares e heurísticas de composição. O sistema é composto por dezesseis módulos e totaliza 1835 linhas de código.

Com a finalidade de permitir a comparação com resultados obtidos por outros autores, é importante frisar que a distância calculada entre cada par de vértices é dada pela distância Euclidiana $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \forall i,\ j\ \in V, i\ \neq j$ sem arredondamentos. Além disso, o número de veículos utilizados deve ser uma variável de entrada do problema no caso de solução por algoritmos exatos, mas pode ser uma variável de decisão no caso de heurísticas/metaheurísticas (SOSA, GALVÃO e GANDELMAN, 2007). O sistema foi processado em um microcomputador PC com processador Intel Core i3 M 350 de 2.27 GHz, e os seguintes resultados computacionais foram obtidos para as quatro primeiras instâncias de teste de Christofides, Mingozzi e Toth, utilizando o *Refset* com dez soluções ($b_1 = 5$ e $b_5 = 5$) e $PSize = 30$, conforme visto na tabela 2:

Tabela 2 – Resultados computacionais das instâncias processadas com o sistema

Inst.	Nº Clientes	Cap. Veíc.	Psize	Solução	T Iniciais (min)	T Total (min)	Distância	Melhor Lit.	DPR
1	50	160	30	31	0,67	6,63	**533,81**	524,61	**1,75%**
2	75	140	30	31	2,46	35,96	**877,55**	835,26	**5,06%**
3	100	200	30	180	5,31	94,87	**873,06**	826,14	**5,68%**
4	150	200	30	401	18,71	787,43	**1092,11**	1028,42	**6,19%**

Fonte: Organizado pelo autor.

A primeira coluna da tabela refere-se ao número da instância; a segunda ao número de clientes contidos na instância; a terceira à capacidade dos veículos dada como restrição para este conjunto de dados; a quarta ao número de soluções iniciais geradas pelo sistema; a quinta ao número da melhor solução encontrada; a sexta ao tempo necessário para a geração das soluções iniciais; a sétima ao tempo total de processamento do algoritmo; a oitava à distância obtida na melhor solução encontrada pelo sistema; a nona à melhor solução disponível na literatura para a instância; e a décima ao Desvio Percentual Relativo (DPR).

Sosa, Galvão e Gandelman (2007) definem o DPR para uma dada instância como sendo o valor da função objetivo encontrado pelo algoritmo implementado menos a melhor

solução disponível na literatura para esta instância, dividida pela melhor solução disponível na literatura para esta instância.

O DPRM, dado pela média dos desvios encontrados, foi de 4,67% comparado aos melhores resultados disponíveis na literatura para as quatro primeiras instâncias de Christofides, Mingozzi e Toth.

Abaixo (Ilustração 3) segue a plotagem de uma das soluções obtidas por meio do sistema, para a terceira instância testada:

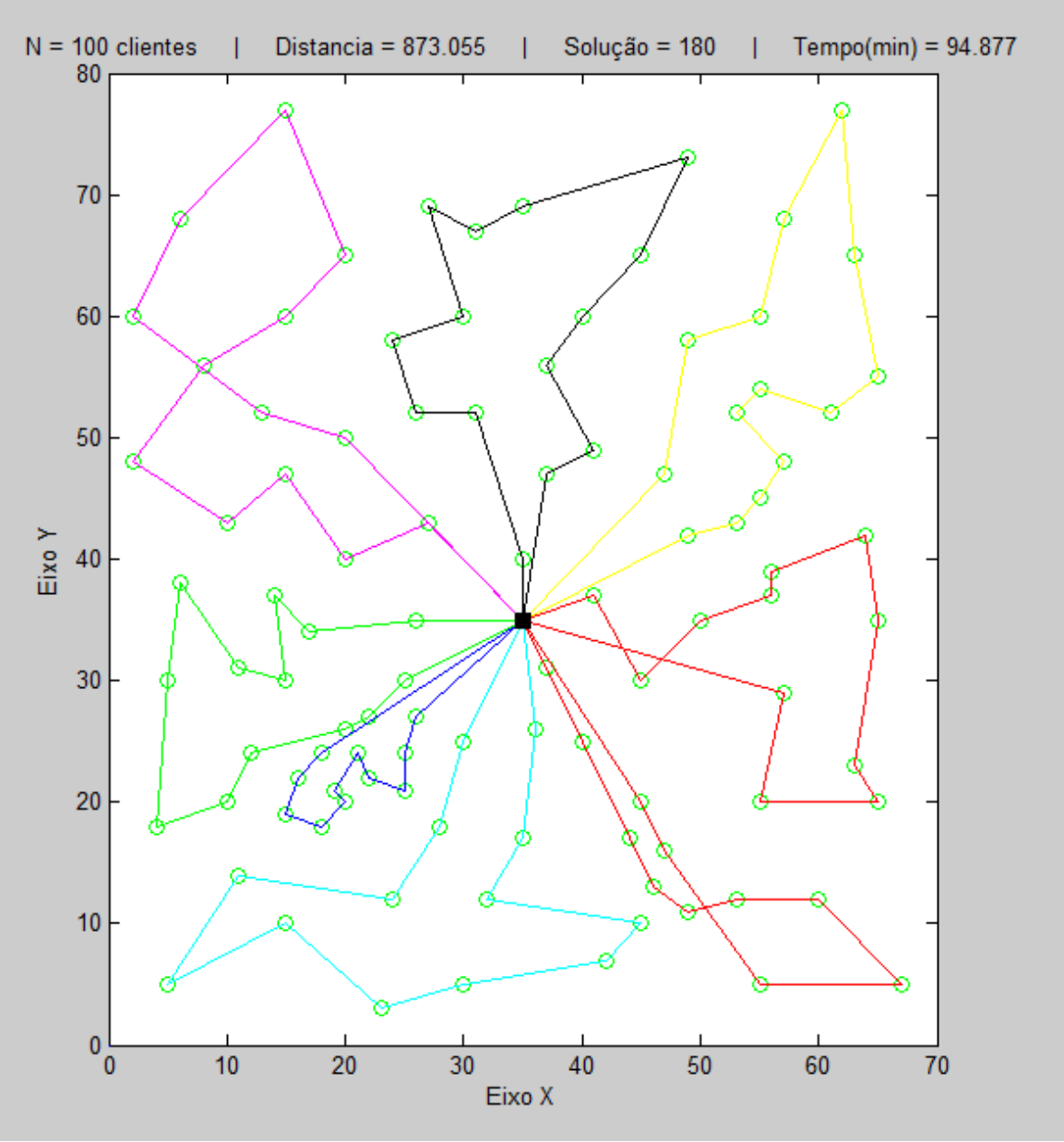

Ilustração 3 – Melhor solução obtida para a terceira instância de dados
Fonte: Organizado pelo autor.

4.2 DISCUSSÃO DOS RESULTADOS

É possível notar, baseado nos resultados obtidos, que a metaheurística da Busca Dispersa teve desempenho computacional relevante para a solução do PRV proposto, obtendo resultados muito próximos aos melhores listados na literatura, com um DPRM de 4,67%. Os resultados melhoram substancialmente com a utilização de um número maior de soluções iniciais (*PSize*), mas o custo computacional envolvido torna este aumento inviável para a maior parte dos conjuntos de dados, sendo obtidos tempos de processamento superiores a 19 horas com $PSize = 50$ para o terceiro conjunto de dados sem se obter uma solução definitiva (critério de parada do sistema, que ocorre quando o *Refset* não é mais atualizado). Portanto, foi definido empiricamente um *PSize* de 30 soluções iniciais para permitir a solução de todas as instâncias em um tempo viável.

Um teste realizado com a primeira instância de Christofides, Mingozzi e Toth utilizando $PSize = 50$ obteve a distância total de 528,49 em 13,93 minutos (aproximadamente o dobro do utilizado para $PSize = 30$), configurando um DPR de 0,74% em relação ao melhor resultado obtido na literatura, conforme pode ser visto na ilustração 4:

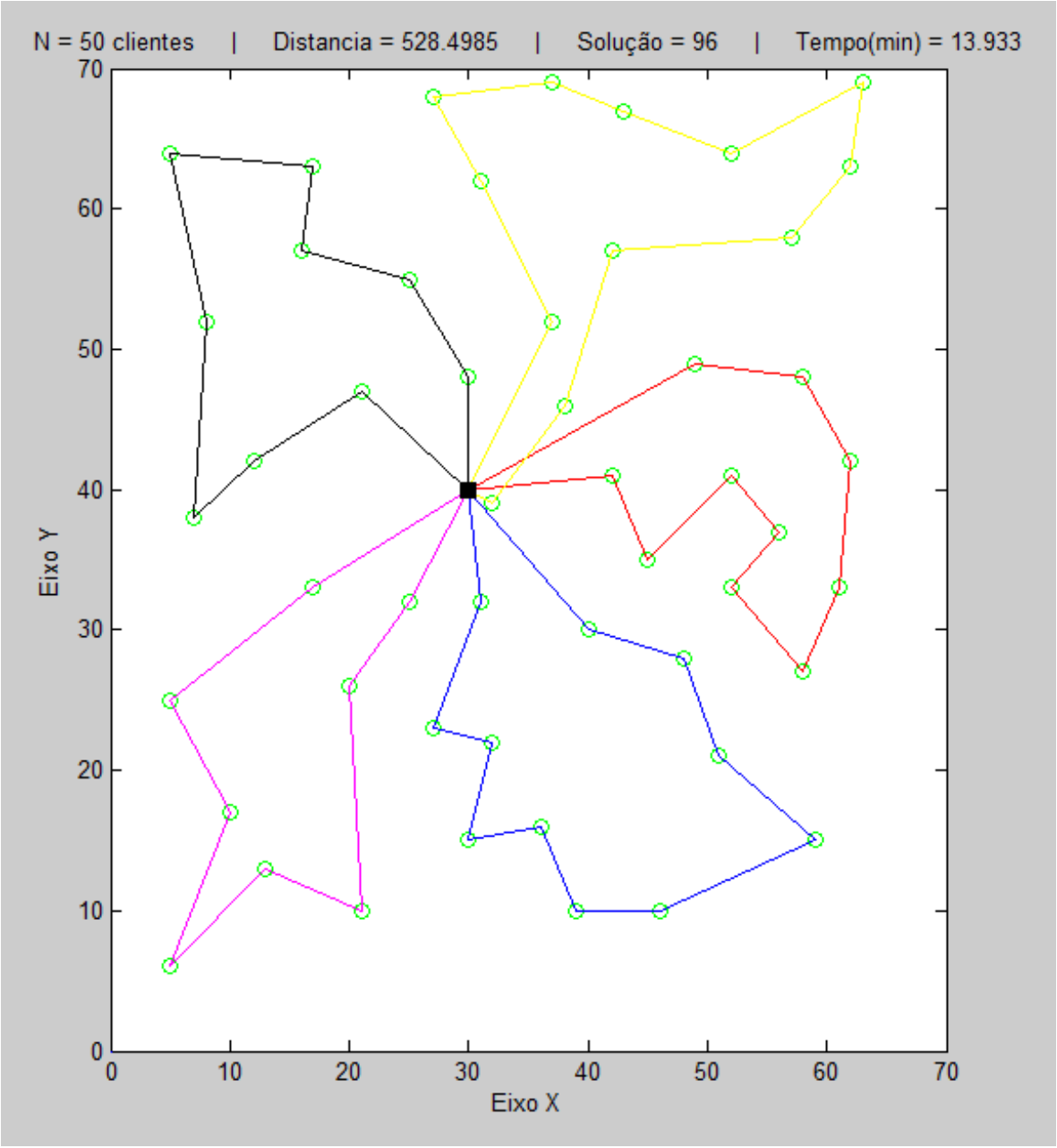

Ilustração 4 – Melhor solução obtida para a primeira instância de dados (PSize = 50)
Fonte: Organizado pelo autor.

O tamanho do *Refset* foi estabelecido com base no artigo de Sosa, Galvão e Gandelman (2007) onde um conjunto com $b = 10$ soluções, sendo $b_1 = 5$ e $b_2 = 5$, obteve os melhores resultados em testes utilizando estas instâncias.

Notou-se também um aumento expressivo do tempo computacional utilizado para processar os conjuntos de dados à medida que o número de vértices aumentava, chegando a 787,43 minutos (aproximadamente 13 horas) de processamento para encontrar a solução final da quarta instância de dados.

5. CONCLUSÕES

Os objetivos do trabalho foram alcançados, com a modelagem matemática e computacional do sistema realizada com êxito e os resultados obtidos compatíveis às expectativas. Cabe um especial destaque ao desempenho obtido no processamento da primeira instância de Christofides, Mingozzi e Toth, com o qual foi obtido um DPR de 1,75% para 6,63 minutos de processamento e 0,74% para 12,64 minutos, demonstrando a capacidade do sistema em encontrar boas soluções viáveis para um número pequeno de vértices. Entretanto, o custo computacional necessário para a obtenção de soluções em instâncias maiores superou muito o esperado, chegando a 13 horas de processamento, o que abre espaço para que este método seja revisado e adaptado para o tratamento de dados contendo um maior número de vértices.

5.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Este trabalho terá a sua continuidade voltada para o desenvolvimento de novas metaheurísticas híbridas e outros procedimentos computacionais que permitam soluções mais eficientes e de menor custo computacional para esta classe de problemas complexos, em especial quando se tem um maior número de vértices envolvidos.

A motivação para esta pesquisa é o alto custo dos softwares de roteamento de veículos existentes no mercado atualmente, que inviabiliza a utilização destes por empresas de pequeno e médio porte. Estas são responsáveis por grande parte dos empregos gerados em nosso país, mas apresentam baixa produtividade (geram 53% dos empregos formais, mas são responsáveis por apenas 20% do PIB Brasileiro, segundo a revista Exame PME). Métodos computacionais mais eficientes permitirão a utilização de sistemas de roteamento a partir de servidores web, com menor custo de instalação e manutenção, democratizando assim o acesso a estas tecnologias que notavelmente trazem grande vantagem competitiva (redução de custos – horas extras, subutilização dos veículos, distâncias percorridas – e responsabilidade ambiental – redução do uso de recursos naturais e emissão de gases poluentes) àqueles que as utilizam.

REFERÊNCIAS

AI, T. J., ; KACHITVICHYANUKUL, V. **A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery**, Computers & Operations Research, n 36, 2009, p.1693-1702

BELFIORE, Patrícia Prado and YOSHIZAKI, Hugo Tsugunobu Yoshida. **Scatter search para problemas de roteirização de veículos com frota heterogênea, janelas de tempo e entregas fracionadas**. *Prod.* [online]. 2006,.16, n.3, p. 455-469.

CHRISTOPHER, Martin, **Logística e gerenciamento da Cadeia de Suprimentos**. 1 ed. São Paulo: Pioneira, 1997

ESPEJO, Luis Gonzalo Acosta ; GALVAO, Roberto D.. **O uso das relaxações lagrangeana e surrogate em problemas de programação inteira.** *Pesqui. Oper.* [online]. 2002, 22, n.3, p. 387-402.

FIGUEIREDO, A. P. S. . **Localização de ambulâncias pelo modelo TEAM - Solução Através do Algoritmo Genético Construtivo**. In: IV Worcap - Workshop dos cursos de computação aplicada do INPE, 2004, São José dos Campos. **Anais** do IV Worcap. São José dos Campos : INPE, 2004.

FIGUEIREDO, A. P. S. . **Localização de ambulâncias: uma aplicação para a cidade de São José dos Campos - SP**. In: Simpósio Brasileiro de Sensoriamento Remoto, 2005, Goiania - GO. Anais do XII Simpósio Brasileiro de Sensoriamento Remoto. São José dos Campos : Inpe, 2005.

FIGUEIREDO, A. P. S. . **Modelos de localização de ambulâncias**. In: Workshop dos Cursos de Computação Aplicada do INPE, 3, 2003, São José dos Campos. III WORCAP. São José dos Campos : INPE, 2003. p. 229-234.

FRAGA, M. C. P. **Uma metodologia híbrida: Colônia de formigas - busca tabu - reconexão por caminhos para resolução do problema de roteamento de veículos com janela de tempo**. Dissertação de Mestrado - Cefet - MG, 2007.

GALVAO, Roberto Diéguez; BARROS NETO, Júlio Francisco; FERREIRA FILHO, Virgílio J. M. ; HENRIQUES, Horácio Brescia de Sousa. **Roteamento de veículos com base em sistemas de informação geográfica**. *Gest. Prod.* [online]. 1997, 4, n.2, p. 159-174.

HILLIER, Frederick S.; LIEBERMAN, Gerald J, **Introdução à pesquisa operacional**. 8 ed. São Paulo: McGraw-Hill, 2006

INSTITUTO DE LOGÍSTICA E SUPPLY CHAIN, 2010, **Panorama logístico - Os nelhores prestadores de serviço logístico e ferrovias do Brasil**, *ILOS – Instituto de Logística e Supply Chain*, Disponível em http://www.ilos.com.br/site/index.php?option=com_content&task=blogcategory&id=14&Itemid=54, (Acesso em: março/2010)

LAPORTE, G. **The vehicle routing problem: an overview of exact and approximate algorithms**. European Jounal of Operational Research, n 59: 345-358, 1992.

OCHI, L. S., TORTELLY JUNIOR, A. **Uma metaheurística híbrida GRASP+tabu para o problema de roteamento periódico de uma frota de veículos**. **Anais** do XXXV Simpósio Brasileiro de Pesquisa Operacional (XXXV SBPO), 2003, Natal, RN.SOBRAPO, 2003. v.1. p.1 – 11

OLIVEIRA, Humberto César Brandão de, **Um modelo híbrido estocástico para tratamento de um problema de roteamento de veículos com janela de tempo**. Tese de M.Sc., CIN/UFPE, Recife, PE, Brasil. 2007

OLIVEIRA, Humberto César Brandão de; VASCONCELOS, Germano Crispim.; **A hybrid search method for the vehicle routing problem with time windows**. Annals of Operations Research Year: 2008

REGO, César; LEÃO, Pedro, **A scatter search tutorial for graph-based permutation problems. Working Paper Series**, Hearin Center for Enterprise Science, University of Mississipi, 2001.

RODRIGUES, Paulo Roberto Ambrosio, **Introdução aos sistemas de transporte no Brasil e à logística internacional**. 4 ed. São Paulo: Aduaneiras, 2007

SILVA MELO, André Cristiano da and FERREIRA FILHO, Virgílio José Martins. **Sistemas de roteirização e programação de veículos**. *Pesqui. Oper.* [online]. 21, n.2, 2001, p. 223-232.

SOSA, Nélida Gladys Maquera; GALVAO, Roberto Diéguez and GANDELMAN, Dan Abensur. **Algoritmo de busca dispersa aplicado ao problema clássico de roteamento de veículos**. *Pesqui. Oper.* [online]. 27, n.2, 2007, p. 293-310.

TAHA, Hamdy A. **Pesquisa Operacional**. 8 ed. São Paulo: Pearson Prentice Hall, 2008.

TIGERLOG – CONSULTORIA E TREINAMENTO EM LOGÍSTICA, 2010, **Curiosidades em Logística**, TIGERLOG – Consultoria e Treinamento em Logística, Disponível em http://www.tigerlog.com.br/logistica/curiosidades_rodoviario.asp, (Acesso em: março/2010)

TOTH, P.; VIGO, D. **The Vehicle Routing Problem**. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, U.S.A, 2002.

VITASEK, Kate, **Supply Chain Management Terms and Glossary**, *Council of Supply Chain Management Professionals*, Disponível em http://cscmp.org/digital/glossary/glossary.asp, (Acessado em: março/2010)

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA

Termo de Declaração de Autenticidade de Autoria

Declaro, sob as penas da lei e para os devidos fins, junto à Universidade Federal de Juiz de Fora, que meu Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia de Produção é original, de minha única e exclusiva autoria. E não se trata de cópia integral ou parcial de textos e trabalhos de autoria de outrem, seja em formato de papel, eletrônico, digital, áudio-visual ou qualquer outro meio.
Declaro ainda ter total conhecimento e compreensão do que é considerado plágio, não apenas a cópia integral do trabalho, mas também de parte dele, inclusive de artigos e/ou parágrafos, sem citação do autor ou de sua fonte.
Declaro, por fim, ter total conhecimento e compreensão das punições decorrentes da prática de plágio, através das sanções civis previstas na lei do direito autoral[1] e criminais previstas no Código Penal[2], além das cominações administrativas e acadêmicas que poderão resultar em reprovação no Trabalho de Conclusão de Curso.

Juiz de Fora, _____ de _____ de 20____.

_____	_____
NOME LEGÍVEL DO ALUNO (A)	Matrícula
_____	_____
ASSINATURA	CPF

[1] LEI N° 9.610, DE 19 DE FEVEREIRO DE 1998. Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.
[2] Art. 184. Violar direitos de autor e os que lhe são conexos: Pena – detenção, de 3 (três) meses a 1 (um) ano, ou multa.