HEURÍSTICAS E METAHEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE COM GRUPAMENTOS

Melise Maria Veiga de Paula

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS.

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc.

Prof. Luiz Satoru Ochi., D.Sc.

Prof. Raul Fonseca Neto, D.Sc.

Prof. Paulo Roberto Oliveira, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 2001

PAULA, MELISE MARIA VEIGA

Heurísticas e Metaheurísticas para o Problema do Caixeiro Viajante com Grupamentos [Rio de Janeiro] 2001

IX, 97p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas, 2001)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Problema do Caixeiro Viajante com Grupamentos
2. GRASP
3. VNS

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

HEURÍSTICAS E METAHEURÍSTICAS PARA O PROBLEMA DO CAIXEIRO VIAJANTE COM GRUPAMENTOS

Melise Maria Veiga de Paula

Dezembro/2001

Orientador: Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

Neste trabalho, são propostos algoritmos heurísticos e metaheurísticos para a solução aproximada de uma generalização do Problema do Caixeiro Viajante (PCV), conhecida na literatura como Problema do Caixeiro Viajante com Grupamentos (PCVG) ou "Clustered Traveling Salesman Problem".

No PCVG, o conjunto de vértices, V, é particionado em M subconjuntos disjuntos, denominados grupamentos: $V = V_1 \cup V_2 \cup ... \cup V_M$. O objetivo do problema é encontrar um ciclo hamiltoniano de custo mínimo de maneira que todos os vértices que pertencem a um mesmo grupamento sejam visitados contiguamente.

São apresentadas várias propostas de algoritmos heurísticos e metaheurísticos utilizando os conceitos de GRASP e VNS.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

HEURISTICS AND METAHEURISTICS FOR CLUSTERED TRAVELING SALESMAN PROBLEM

Melise Maria Veiga de Paula

December/2001

Advisor: Nelson Maculan Filho

Department: System Engineering

This work proposes heuristic and metaheuristic algorithms for an approximate solution of the Clustered Traveling Salesman Problem (CTSP), which is a well-known generalization of the Traveling Salesman Problem. In the CTSP, the set of nodes, V, is partitioned into M disjoint subsets called clusters, $V = V_1 \cup V_2 \cup ... \cup V_M$. The goal of the CTSP is to find the tour with minimum cost in which all the nodes of the same cluster can be visited in a contiguous manner. We present several heuristic and metaheuristic algorithms based on GRASP and VNS.

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Problemas de otimização combinatória caracterizam-se, normalmente, por possuírem um espaço grande, porém finito, de soluções. Entretanto, a resolução desses problemas através de uma enumeração completa das soluções é praticamente inviável, visto que, o número de soluções possíveis, na maioria dos casos, cresce exponencialmente com o tamanho do problema. Um outro fator que merece ser lembrado é que muitos destes problemas são computacionalmente intratáveis ou suficientemente grandes para impedir ou limitar a aplicação exclusiva de algoritmos exatos para resolvê-los.

Diante dessas duas dificuldades intrínsecas aos problemas de otimização combinatória, torna-se clara a necessidade de se obter ferramentas que permitam oferecer soluções rápidas para problemas com dimensões reais. Os métodos heurísticos são comumente aplicados na resolução de tais problemas com o intuito de obter soluções de boa qualidade. Entende-se por soluções de boa qualidade como aquelas que, apesar de não corresponderem, necessariamente, ao ótimo, representam bons limitantes (inferior ou superior) para um dado problema.

Contudo, métodos heurísticos tradicionais de construção e/ou construção e busca local possuem algumas limitações históricas como, por exemplo, a convergência prematura para um ótimo local, muitas vezes, distante de um ótimo global em problemas de otimização.

A partir dos anos 1980, começaram a surgir, na literatura, propostas de heurísticas genéricas com o intuito de se tentar superar algumas das limitações das heurísticas tradicionais. Estes métodos, conhecidos como Metaheurísticas ou Heurísticas Inteligentes, possuem ferramentas que tentam, por exemplo, superar as armadilhas de

uma parada prematura em ótimos locais ainda distantes de um ótimo global. Podemos citar algumas das Metaheurísticas mais conhecidas: Busca Tabu, Algoritmos Genéticos, Simulated Annealing, GRASP (Greedy Randomized Adaptive Search Procedures) e VNS (Variable Neighborhood Search).

Um problema de otimização combinatória clássico e muito conhecido na literatura é o Problema do Caixeiro Viajante (PCV). Muitos problemas reais podem ser modelados como um PCV e, desta forma, serem solucionados obtendo boas e, em alguns casos, ótimas soluções.

Neste trabalho, será abordada uma das generalizações do PCV denominada Problema do Caixeiro Viajante com Grupamentos (PCVG). Uma das razões que motivaram o nosso estudo vem da diversidade de classes de problemas reais que podem ser modeladas como um PCVG. Entre elas podemos citar problemas de escalonamento, problemas de roteamento e outras aplicações onde os serviços estabelecem prioridades de execução.

Além disso, o PCVG, por ser uma generalização do PCV, pertence à classe NP-Árdua, limitando o uso exclusivo de técnicas exatas. Desta forma, surge a necessidade de se desenvolver procedimentos heurísticos ou aproximados para a sua solução.

A proposta deste trabalho é desenvolver algoritmos heurísticos para o PCVG utilizando conceitos das metaheurísticas GRASP e VNS que têm produzido soluções competitivas, quando comparadas com outras metaheurísticas, em diferentes problemas de otimização combinatória.

Esta dissertação está dividida da seguinte forma: no capítulo 2, é dada uma visão geral do Problema do Caixeiro Viajante com Grupamento (PCVG), onde são apresentadas as diversas variações do PCVG e uma descrição formal do problema. No capítulo 3, é apresentada a revisão da literatura disponível para o problema. Os

conceitos básicos das metaheurísticas GRASP e VNS são abordados no capítulo 4. No capítulo 5, são descritas as heurísticas propostas para solucionar o problema. Já no capítulo 6, são descritos os algoritmos propostos baseados no GRASP e no VNS. Os resultados computacionais são apresentados no capítulo 7 e a conclusão no capítulo 8. Finalmente, o capítulo 9 apresenta as referências bibliográficas usadas nesse trabalho.

2. Problema do Caixeiro Viajante com Grupamentos

Neste capítulo, é apresentada uma descrição do Problema do Caixeiro Viajante com Grupamentos (PCVG) ou "Clustered Traveling Salesman Problem" (CTSP). Formalmente, o PCVG pode ser definido como se segue:

Seja G = (V,E) um grafo completo e não direcionado com V sendo o conjunto de vértices e E = $\{(v_i,v_j)$, onde v_i é diferente de v_j e $v_i, v_j \in V\}$ sendo o conjunto de arestas de forma que, cada aresta (v_i,v_j) está associada a um número $d_{ij} \in R^+$ que será chamado de distância entre v_i e v_j. Considere ainda que, o conjunto de vértices, V, é particionado em M conjuntos, V_1, V_2, ..., V_M, denominados grupamentos. Cada vértice v_i deve pertencer a um, e somente um, grupamento do grafo, ou seja, os M grupamentos do problema são M subconjuntos disjuntos do conjunto de vértices V, portanto:

$V_1 \cup V_2 \cup ... \cup V_M = V$ e $V_1 \cap V_2 \cap ... \cap V_M = \varnothing$

O objetivo é encontrar um ciclo hamiltoniano de custo mínimo em G de modo que, todos os vértices que pertencem a um mesmo grupamento sejam visitados contiguamente.

O Problema do Caixeiro Viajante (PCV) clássico pode ser visto como um caso especial do PCVG onde M é igual a 1. Sendo assim, podemos afirmar que o PCVG é NP-Árduo.

Segundo CHISMAN (1975), uma interpretação do PCVG pode ser:

"Um caixeiro viajante deve partir de uma cidade, denominada origem ou depósito, visitar cada cidade restante do problema uma única vez e retornar à cidade origem percorrendo a menor distância possível, como no PCV. Além disso, o conjunto de cidades é dividido em subconjuntos disjuntos de modo que, o caixeiro deve visitar as cidades de cada subconjunto contiguamente."

Neste contexto, as cidades são representadas pelos vértices do grafo e cada d_{ij} representa o tempo, ou o custo ou a distância do deslocamento entre as cidades i e j, representadas pelos vértices v_i e v_j, respectivamente. As figuras abaixo ilustram um exemplo do problema:

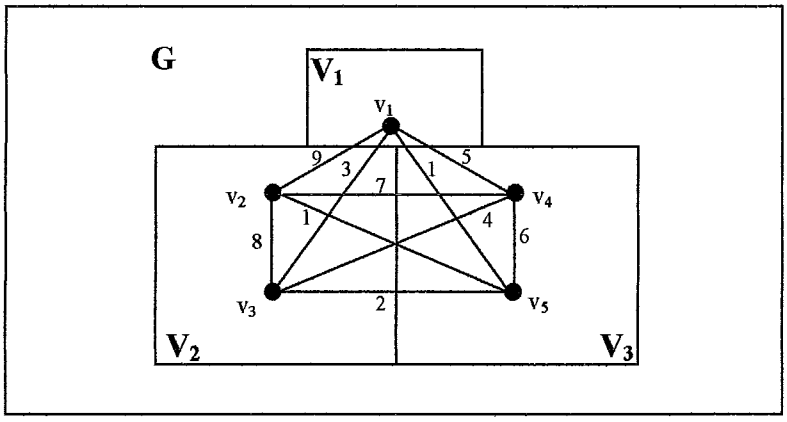

Figura 2.1 - $G = (V_G, E_G)$: Instância do PCVG com 3 grupamentos

No grafo representado pela Figura 2.1 pode ser definido um PCVG onde:

- M = {número de grupamentos de G} = 3
- V_G = {conjunto de vértices} = $V_1 \cup V_2 \cup V_3$
- $V_1 = \{v_1\}$, $V_2 = \{v_2, v_3\}$ e $V_3 = \{v_4, v_5\}$
- $V_G = \{v_1\} \cup \{v_2, v_3\} \cup \{v_4, v_5\} = \{v_1, v_2, v_3, v_4, v_5\}$
- E_G = {conjunto de arestas} =

  $\{(v_1,v_2), (v_1,v_3), (v_1,v_4), (v_1,v_5), (v_2,v_3), (v_2,v_4), (v_2,v_5), (v_3,v_4), (v_3,v_5), (v_4,v_5)\}$

Observe que, se as restrições de contiguidade dos vértices em cada grupamento for desconsiderada, pode-se definir, em G, um PCV.

A seguir serão apresentadas duas soluções para o PCVG definido no grafo da Figura 2.1. A Figura 2.2 representa uma solução viável para a instância do PCVG definida no grafo G da Figura 2.1, isto é, todos os vértices de cada grupamento são visitados contiguamente. Na Figura 2.3, a solução apresentada é viável para o PCV definido no

grafo G, porém, é inviável para o PCVG, pois os vértices dos grupamentos V_2 e V_3 não são visitados contiguamente.

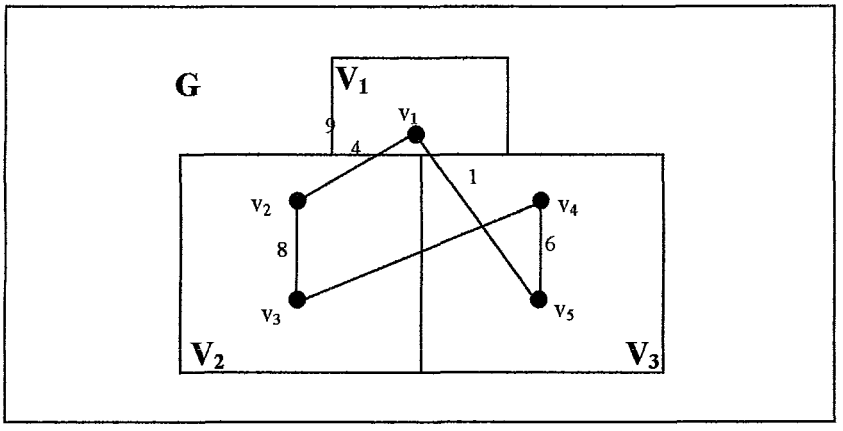

Figura 2.2 - Uma solução viável para o PCVG definido na Figura 2.1

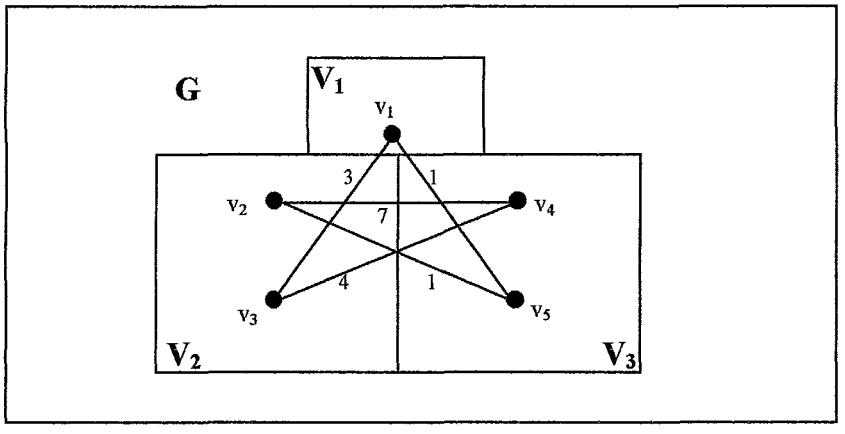

Figura 2.3 - Uma solução inviável para o PCVG, mas viável para o PCV, definidos na Figura 2.1, desconsiderando a condição de contiguidade

O PCVG pode ser dividido em dois subproblemas: o problema definido a nível dos vértices do grafo que define o PCVG, denominado, Problema Inter-Grupamento (PIG) e o problema definido a nível dos grupamentos, denominado Problema Entre-Grupamento. No PIG, o objetivo é encontrar a seqüência de visitas entre os vértices do grafo. Enquanto que, o PEG consiste em encontrar a seqüência ótima entre os grupamentos definidos no conjunto de vértices do grafo. Os dois subproblemas, apesar de distintos, são dependentes entre si, ou seja, a solução do problema Inter_Grupamento interfere diretamente na solução do Problema Entre_Grupamento e vice-versa.

No caso geral do Problema do Caixeiro Viajante com Grupamento, também chamado Problema do Caixeiro Viajante com Grupamento Irrestrito – PCVGI (Free Clustered Traveling Salesman Problem - FCTSP) (CHISMAN, 1975, JONGENS *et al.*, 1985), a ordem de visitas entre os grupamentos não é definida. Portanto, faz parte da solução do PCVGI, resolver os problemas Inter e Entre-Grupamento.

Além disso, no PCVG, o vértice inicial pode, ou não, ser um dado do problema. Neste trabalho será considerado o PCVGI sem vértice inicial estabelecido. Nos próximos capítulos, o PCVGI sem vértice inicial estabelecido será chamado de PCVG.

Um caso especial do PCVG é o Problema do Caixeiro Viajante com Grupamento Restrito-PCVGR (Restricted Clustered Traveling Salesman Problem–RCTSP) (GENDREAU *et. al.*, 1996a, GENDREAU *et. al.*, 1996b, GENDREAU *et. al.*, 1997) também chamado de Problema do Caixeiro Viajante com Grupamento Ordenado (Ordered Clustered Traveling Salesman Problem – OCTSP) (ANILY *et al*,1998) em que a ordem de visita dos grupamentos é predefinida. Neste caso, a solução do problema Entre-Grupamento faz parte dos dados do PCVG, sendo necessário solucionar apenas o problema Inter-Grupamento.

Entre as aplicações do PCVG, pode-se citar *automated warehouse routing* (CHISMAN, 1975, LOKIN, 1978), planejamento de producão e sequenciamento de operações em máquinas controladas numericamente (LOKIN, 1978). Outras aplicações envolvem sistemas de serviços reais, onde os consumidores possuem prioridades de serviço diferentes (GENDREAU *et. al.*, 1996a, GENDREAU *et. al.*, 1996b, GENDREAU *et. al.*, 1997).

3. Literatura existente para o Problema do Caixeiro Viajante com Grupamentos

Estudando um problema real de armazenamento, em CHISMAN(1975), o autor definiu o PCVG e propôs um método de solução para o problema que será descrito a seguir:

Considere G = (V,E) um grafo completo e simétrico onde:

- V: conjunto de vértices de G
- $E = \{(v_i,v_j)$ com v_i diferente de v_j e $v_i, v_j \in V\}$: o conjunto de arestas de G
- $d_{ij} \in R^+$: custo da aresta $(v_i,v_j) \in E$
- $L = \max \{d_{ij}\} \forall$ aresta $(v_i,v_j) \in E$
- $PCVG_G$: instância do PCVG definida em G

Para resolver o PCVG, Chisman propôs o seguinte método:

i) Cria-se uma instância do Problema do Caixeiro Viajante (PCV) baseado na $PCVG_G$ da seguinte forma:

Adicione L ao custo de toda aresta que conecta vértices que não pertencem a um mesmo grupamento, tomando o cuidado de não alterar os valores dos custos das arestas entre vértices de um mesmo grupamento. A instância do PCV, resultante da modificação, é denominada PCV_G.

ii) Resolva o PCV_G aplicando um dos métodos de solução disponíveis para o PCV na literatura.

iii) Depois de encontrada a solução do PCV_G, subtraia L do custo de todas as arestas entre vértices que não pertencem a um mesmo grupamento, obtendo uma solução para o $PCVG_G$ que é a instância original do PCVG.

Para resolver o PCV_G, em CHISMAN (1975), o autor sugere a aplicação de um dos algoritmos de Branch e Bound disponíveis para o PCV. Para mais detalhes desses algoritmos, veja em EASTMAN (1958), LITTLE *et al.* (1963) e HELD (1971).

Para justificar o método, CHISMAN (1975) afirma que:

a) Com a modificação proposta na instância original do PCVG, os custos associados às arestas entre grupamentos diferentes são penalizados. Portanto, na solução encontrada, o número dessas arestas é o mínimo possível, fazendo com que a restrição de contiguidade nos grupamentos seja respeitada.
b) Como os valores dos custos das arestas entre vértices do mesmo grupamento não são alterados, pode-se afirmar que este procedimento não interfere na escolha das arestas entre esses vértices.
c) Uma vez que, o custo das arestas que conectam vértices de grupamentos diferentes são penalizados adicionando um mesmo valor, a escolha dessas arestas não é influênciada pela modificação do PCVG.

CHISMAN (1975) afirma também que o método apresentado neste artigo, quando aplicado ao PCVG, encontrou solução ótima para vários problemas de configurações e tamanhos diferentes.

Como mencionado no capítulo anterior, um caso particular do PCVG é o Problema do Caixeiro Viajante com Grupamento Restrito – PCVGR também chamado Problema do Caixeiro Viajante com Grupamentos Ordenados–PCVGO (ANILY *et al*,1999), em que a ordem de visita dos grupamentos é pré-definida, fazendo parte dos dados do problema.

Em GENDREAU *et al.* (1996b), os autores apresentam três heurísticas, uma para o PCVG e duas, para resolver o PCVGO.

O método proposto para resolver o PCVG, em GENDREAU *et al.* (1996b), foi baseado na proposta de CHISMAN (1975) e denominado H1. Neste método, a instância do PCV (PCV_G) é gerada a partir da instância original do PCVG adicionando uma constante L ($L >> \max\{d_{ij}\}$) aos custos das arestas entre vértices de grupamentos diferentes. Em H1, o PCV_G, resultante do PCVG, é solucionado usando a heurística GENIUS.

GENIUS é uma heurística desenvolvida por GENDRAU *et. al.* (1992) para resolver o PCV e consiste de uma fase de construção e uma fase de otimização (Busca Local). Na fase de construção, denominada GENI (Generalized Insertion Procedure), é construída uma solução viável para o problema. Na fase de otimização, denominada US (Unstringing e Stringing), o objetivo é melhorar a solução construída na fase anterior. GENIUS é descrita detalhadamente no capítulo 5 deste trabalho.

Depois de resolvido o PCV_G, é subtraído L de todos os custos das arestas que pertencem à solução encontrada para o PCV_G e que conectam dois vértices de grupamentos diferentes obtendo uma solução válida para a instância original.

Em GENDREAU *et. al.* (1996a), os autores consideram o PCVGO onde M = 3, denominado Problema do Caixeiro Viajante com Backhauls (PCVB).

No PCVB, os vértices de V são particionados em $\{\{v_1\}, L, B\}$. O vértice v_1 é denominado depósito, L corresponde ao conjunto de vértices que representam os consumidores *Linehaul* e, B, o conjunto de vértices que representam os consumidores *Backhauls*. O PCVB consiste em encontrar um ciclo hamiltoniano de custo mínimo no grafo definido pelos vértices $\{\{v_1\} \cup L \cup B\}$, de modo que, todos os vértices de L

sejam visitados contiguamente, imediatamente depois de v_1 (depósito), seguidos pelos vértices de B e retornando a v_1.

Uma generalização muito popular do PCVB é o Problema de Roteamento de Veículo com Backhauls (PRVB) estudado por DEIF *et al.* (1984), GOLDEN *et al.* (1985), CASCO *et al.* (1988), GOETSCHALCKX *et al.* (1989) e GELINAS *et al.* (1992). No PRVB existe uma demanda q_i associada a cada vértice v_i e uma frota homogênea de m veículos, com capacidade Q, para atender as necessidades dos clientes.

O PRVB consiste em determinar m rotas de veículos de custo total mínimo tais que, (1) cada rota comece e termine em um nó chamado de depósito v_1, (2) cada vértice de $V/\{v_1\}$ deve ser visitado somente uma vez e somente por um veículo, (3) a demanda total de qualquer rota não pode ser maior que Q e (4) em todas as rotas, os consumidores *Backhauls* devem ser visitados contiguamente depois dos consumidores *Linehaul*.

O PCVB pode ser considerado um subproblema do PRVB, no caso particular onde $m = 1$.

Em GENDREAU *et al.* (1996a) são apresentados seis métodos de solução para o PCVB. Além disso, é descrito um limite inferior para o problema.

O primeiro método apresentado por GENDREAU *et al.* (1996a) é o H1, descrito anteriormente.

No segundo método, denominado H2, os autores aplicaram GENIUS separadamente nos vértices de L (conjunto de consumidores Linehaul) e nos vértices de B (conjunto de consumidores Backhauls) encontrando um ciclo com os vértices do conjunto L (T_L) e um ciclo com os vértices do conjunto B (T_B). Sendo que, T_L e T_B consideram todos os vértices dos conjuntos L e B, respectivamente.

Através de modificações nos ciclos T_L e T_B, é obtido um novo ciclo, denominado T_G, que é solução do PCVG. T_G é elaborado da seguinte forma:

Considere $\{v',u'\} \in L$ e $\{v'',u''\} \in B$, tal que, em T_L, exista a aresta (v',u') e, em T_B, exista a aresta (v'',u''). Remova (v',u') de T_L, (v'',u'') de T_B e insira as arestas (v',v''), (v_l,u') e (v_l,u''). A Figura 3.1 ilustra a maneira como T_G é obtido

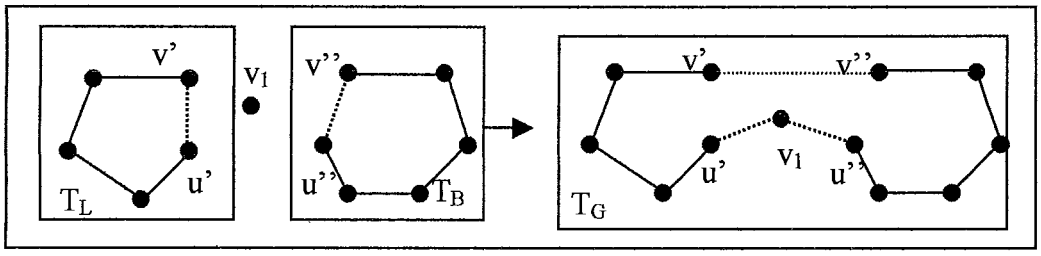

Figura 3.1 – Método H2: Construção de T_G

Cada combinação dos vértices v' e v'' resulta em um novo ciclo T_G. Portanto, depois de consideradas todas as combinações de v' e v'', a solução de menor custo é escolhida como solução do problema.

A seguir, será descrito o terceiro método proposto por GENDREAU *et al.* (1996a), denominado H3.

Assim como no H2, são obtidos dois ciclos T_L e T_B que modificados resultam em um novo ciclo, T_G, que é solução do problema. Porém, no H3, T_B, T_L e T_G são construídos de maneira diferente.

O ciclo T_L é obtido aplicando GENIUS nos vértices do conjunto $\{L \cup \{v_l\}\}$. Já o ciclo T_B é obtido aplicando GENIUS nos vértices do conjunto $\{B \cup \{v_l\}\}$.

O ciclo T_G, obtido a partir de T_L e T_B, é construído da seguinte maneira:

Considere $v' \in \{v_i,v_j\}$ e $v'' \in \{v_s,v_t\}$, onde (v_l,v_i) e (v_l,v_j) são arestas de T_L e (v_l,v_s) e (v_l,v_t) são arestas de T_B. Remova (v_l,v') em T_L e (v_l,v'') em T_B, insira (v',v'').

A Figura 3.2 ilustra a construção de T_G.

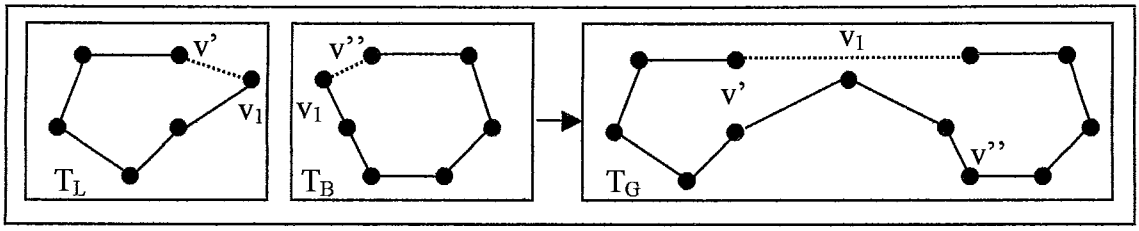

Figura 3.2 – Método H3: Construção de T_G

O número de soluções construídas em H3 é igual ao número de combinações possíveis de v' e v''. A solução de menor custo é escolhida como solução do problema.

O próximo método apresentado em GENDREAU *et al.* (1996a), H4, é descrita a seguir: Começando com um ciclo, denominado S, consistindo do vértice origem, v_1, um vértice de L e um vértice de B, escolhidos aleatoriamente, insira os demais vértices do grafo em S, usando a heurística da Inserção mais Barata e considerando a restrição de contigüidade dos vértices de L e B e a ordem de precedência dos conjuntos.

A Heurística da Inserção mais Barata é descrita com detalhes no capítulo 5 deste trabalho.

No ciclo S, construído na etapa anterior, aplique o procedimento de otimização US, considerando a restrição de contigüidade dos vértices de L e B e a relação de precedência dos conjuntos.

GENDREAU *et al.* (1996a) também apresentaram o método H5 onde a solução do PCVG é obtida da seguinte forma:

Obtenha uma instância do PCV através da instância original do PCVG, modificando a matriz de custo do grafo como em H1. Aplique o procedimento GENI para construir um ciclo inicial S. Em S, aplique o procedimento de pós-otimização Or-OPT, descrito pelos autores.

A solução obtida é uma solução do PCV e deve, portanto, ser modificada para que se torne uma solução do PCVG.

Finalmente, os autores apresentam a heurística H6, onde a solução inicial é construída como em H4. Posteriormente, na fase de otimização, o procedimento Or-OPT é aplicado na solução inicial obtendo uma nova solução para o problema.

Os autores realizaram testes com os seis métodos em instâncias com 100, 200, 300, 500 e 1000 vértices. Para as instâncias com 100, 200 e 300 nós, os resultados foram comparados com o valor do limite inferior do problema que foi definido pelos autores. Em termos de qualidade da solução, H1 é o melhor método seguido por H4, H2, H3, H5 e H6. Na análise dos resultados é possível perceber que os melhores métodos são aqueles baseados em GENIUS. Por outro lado, os piores resultados foram obtidos com a aplicação de H6, baseada na Heurística de Inserção mais Barata e o método Or-Opt. Finalmente, comparando H4 com H6 e H5 com H1, que se diferenciam pela fase de refinamento da solução, é possível observar que US apresenta uma superioridade em relação ao procedimento Or-OPT na fase de refinamento.

Antes de descrever os outros métodos encontrados na literatura do PCVG, será apresentada a definição de alguns conceitos: *Garantia de Performance*, também chamada de *Razão de Aproximação*, e *Algoritmo Aproximado*. A *Garantia de Performance* de um algoritmo para um problema de minimizaçao (ou maximização) é α se é garantido que o algoritmo encontra soluções cujo valor é no máximo (ou no mínimo para problemas de maximização) α vezes o valor da solução ótima. Sendo assim, um *Algoritmo α-aproximado* para um determinado problema é um algoritmo que, executado em tempo polinomial, possui performance garantida igual a α.

ANILY *et al.* (1999b) apresentaram um algoritmo 5/3-aproximado para o PCVGO baseado na Heurística de Christofides (CHRISTOFIDES, 1976) com complexidade $O(n^3)$. Já em ARKIN *et al.* (1997), os autores apresentaram um algoritmo 3.5-

aproximado. Ainda para o PCVGO, o algoritmo apresentado por GENDREAU *et. al.* (1996b) possui razão de aproximação igual a 2.

O próximo trabalho apresentado foi descrito por GUTTMAN *et al.* (2000), onde os autores consideraram 5 variações do PCVG e apresentaram algoritmos de aproximação com garantia de perfomance limitada para resolvê-las. Entre as variações consideradas, quatro se diferenciam na especificação dos vértices de entrada e saída em cada grupamento. Na primeira variação do PCVG ($PCVG_1$), os vértices de entrada e saída em cada grupamento são especificados e os autores descreveram um algoritmo com garantia de performance igual a 1.9091. Na segunda, ($PCVG_2$), dois vértices de cada grupamento são especificados, entretanto, a escolha de qual vértice será a entrada e saída em cada grupamento faz parte da solução do problema. Para essa variação, o algoritmo apresentado possui razão de aproximação igual a 1.8. No terceiro caso, ($PCVG_3$), é dado somente o vértice de entrada de cada grupamento e o algoritmo proposto é 2.643-aproximado. Para o quarto problema considerado, ($PCVG_4$), a escolha dos vértices de entrada e saída de cada grupamento faz parte da solução do problema e o algoritmo apresentado possui razão de aproximação igual a 2.75. Finalmente, os autores apresentaram um algoritmo 5/3-aproximado para o PCVG onde o número de grupamentos é constante.

Dentre as variações do PCVG consideradas em GUTTMAN *et al.* (2000), o $PCVG_4$ é o problema estudado neste trabalho e, a seguir, será dada uma breve descrição do método de solução proposto para o $PCVG_4$.

Na elaboração dos algoritmos que resolvem as variações do PCVG, GUTTMAN *et al.* (2000) usaram algoritmos aproximados elaborados para solucionar problemas conhecidos na literatura como o Problema do Caminho do Caixeiro Viajante (PCCV) (HOOGEVEEN, 1991), Problema do Staker Crane (PSC) (FREDERICKSON, 1978,

JOHNSON, 1985) e Problema do Carteiro Rural (PCR) (FREDERICKSON, 1979, EISELT, 1995, JANSEN,1992).

No método proposto por GUTTMAN *et al.* (2000) para o $PCVG_4$, são construídas duas soluções e a melhor delas é selecionada.

A seguir é dada uma breve descrição do algoritmo para a versão do PCVG sem especificação dos vértices de entrada e saída dos grupamentos:

Dados de Entrada:

1. Grafo G = (V,E), onde V é o conjunto de vértices e E, o conjunto de arestas
2. $V_1, V_2, ..., V_M$: M subconjuntos disjuntos de V

Inicio

1. Construção da Solução S_1

   1.1- Em cada grupamento V_I, considere o problema de encontrar um caminho hamiltoniano de custo mínimo entre os vértices do grupamento, este problema é uma variação do Problema do Caminho do Caixeiro Viajante (PCCV) onde os vértices inicial e final do caminho não são especificados. Os autores aplicaram um algoritmo desenvolvido para o PCCV encontrando um caminho P_I em cada grupamento V_I.

   1.2- Deixe s_I e t_I ser os vértices inicial e final no caminho P_I, encontrado anteriormente. Nesta etapa, os autores propõem uma alteração no grafo substituindo o caminho encontrado em cada grupamento V_I por uma aresta entre os vértices (s_I,t_I) com o mesmo peso do caminho P_I. O problema modificado ($PCVG_M$) pode ser considerado como um problema do carteiro rural (PCR), onde o conjunto de arestas especiais (E') é composto das arestas (s_I,t_I) acrescentadas nesta etapa. No grafo modificado, os autores aplicaram um algoritmo desenvolvido para o PCR encontrando uma solução S'.

1.3- Na solução encontrada, S', cada aresta (s_I,t_I) foi substituída pelo caminho hamiltoniano P_I, obtendo S_1.

2. Construção da Solução S_2

2.1- Em cada grupamento, encontrar os vértices s_I e t_I tal que, a distância entre s_i e t_I seja a máxima possível. Esses vértices, s_I e t_I, serão os vértices de entrada e saída em cada grupamento V_I do grafo.

2.2- Aplicar o algoritmo proposto para a versão do PCVG onde são especificados os vértices de entrada e saída de cada grupamento encontrando uma solução (S_2) para o problema original.

3. Escolha a melhor solução entre S_1 e S_2 para ser a solução do problema.

Para esta variação do PCVG, GUTTMAN *et al.* (2000) afirmam que o algoritmo proposto possui garantia de performance igual a 2.75.

Em uma outra classe de métodos encontrados na literatura do PCVG, foram desenvolvidos algoritmos genéticos para solucionar o PCVG.

Em POTVIN *et al.* (1995) é descrito um algoritmo genético para o PCVGO, onde o processo de reprodução é feito em dois níveis: a nível dos grupamentos, para determinar a seqüência de visita entre os grupamentos e a nível dos vértices, para determinar a seqüência de visita entre os vértices. A nível dos vértices, é utilizada a heurística genética ERX (Genétic Edge Recombination) que é um operador desenvolvido por WHITLEY *et al* (1989) para gerar a seqüência de visita entre os vértices de cada grupamento. A nível dos grupamentos, são utilizados operadores clássicos para geração de cromossomos filhos, respeitando a ordem preestabelecida de visita entre os grupamentos. Neste trabalho, os autores compararam o algoritmo genético com uma

implementação do método H1, descrito anteriormente. Os resultados computacionais mostraram que, em termos de qualidade de solução, o algoritmo genético mostra uma superioridade em relação à H1, porém H1 é mais rápida.

Em POTVIN *et. al.* (1996) é descrito um algoritmo genético para o PCVG. Assim como no trabalho anterior, no método proposto pelos autores, a reprodução é feita a nível dos grupamentos, para determinar a seqüência de visitas entre os grupamentos e a nível dos vértices, encontrando a ordem de visita dos vértices de cada grupamento. Porém, os autores usam o operador ERX tanto na reprodução a nível de grupamentos quanto a nível de vértices. O operador de mutação foi o procedimento de busca local 2-opt, definido por LIN *et al* (1973), que foi aplicado primeiro a nível de grupamento e depois, em cada grupamento, a nível de vértices. Para testar o método, os autores usaram dois tipos diferentes de grafos: grafo aleatório e grafo geométrico. No grafo aleatório, os vértices foram gerados aleatoriamente em um quadrado de dimensões $[0,100]^2$ distribuídos uniformemente. Além disso, os vértices eram associados aleatoriamente aos grupamentos na ordem em que eram gerados. No outro tipo de grafo, grafo geométrico, é definido um quadrado de dimensões iguais a $[0,100]^2$ e o ponto [50,50] que é considerado como o vértice origem do problema. Suponha que M seja o número de grupamentos do problema, o quadrado é dividido em M retângulos iguais que correspondem aos grupamentos do problema. Finalmente, os vértices são gerados aleatoriamente em cada retângulo segundo uma distribuição uniforme. Os problemas foram criados com número de vértices iguais a 100, 300 e 500 e com número de grupamentos iguais a 4 e 10.

Como no artigo descrito anteriormente, os autores implementaram a heurística H1 e compararam os resultados de H1 com aqueles obtidos pela aplicação do AG. Foram apresentados os custos das soluções, o tempo de execução obtido por cada método e a

razão entre o valor das soluções encontradas e o valor do limite inferior do problema descrito por JONGENS *et al.* (1985).

Observando os resultados, os autores puderam constatar que AG é mais rápido em problemas onde o número de grupamento é maior, o que se justifica pelo fato de que, nesses casos, o problema de otimização em cada grupamento é menor. Além disso, as soluções obtidas por AG são melhores que H1, porém AG requer um tempo de execução maior que H1.

Quando são comparados os custos das soluções com os limites inferiores de cada problema, igualmente, AG apresenta uma razão menor entre os custos das soluções e o limite inferior, quando comparado com a razão entre os custos das soluções obtidas por H1 e o mesmo limite.

Outro algoritmo genético para o PCVG foi desenvolvido por RABELO *et. al* (1997). Os autores desenvolveram uma adaptação do operador ERX para solucionar o problema a nível dos vértices e a nível dos grupamentos, ou seja, a adaptação do operador ERX foi usada tanto para determinação das sequências entre os vértices de cada grupamento, quanto para determinação da sequência de visita entre os grupamentos. Como refinamento, foi usada a heurística 2-opt (LIN *et. al.*, 1973). O algoritmo foi comparado com uma implementação da heurística H1. Com os resultados obtidos, RABELO *et al.* (1997) observaram que, como no caso anterior, o algoritmo genético implementado obteve melhores soluções que H1.

4. GRASP e VNS

4.1. GRASP - Procedimento de Busca Adaptativa Aleatória Gulosa

GRASP (Greedy Randomize Adaptive Search Procedures) é uma metaheurística desenvolvida por T. Feo e M. Resende (FEO *et. al.*,1995) que consiste em um procedimento iterativo, combinando um método construtivo com uma busca local. Cada iteração do GRASP é constituída de duas fases: uma fase de construção em que se constrói, elemento por elemento, uma solução viável para o problema, e uma fase de busca local, quando se aplica uma técnica de busca na solução construída na fase anterior, com o objetivo de encontrar um ótimo local. A melhor solução obtida na fase de busca local é comparada às soluções das iterações anteriores e a melhor solução é mantida como resultado.

Em FEO *et. al.* (1995) e RESENDE (1998), os autores apresentam o método descrevendo seus componentes básicos. Uma revisão detalhada da literatura de GRASP pode ser vista em FESTA et al. (2000), onde os autores afirmam que GRASP foi aplicado obtendo boas soluções em diversas classes de problemas de otimização combinatória, tais como: Problemas de Escalonamento (BARD *et al.* (1989) e FEO *et al.* (1991)), Problemas de Planarização em Grafos (RESENDE *et al.*, 1997, RESENDE *et al.*, 2001), Problemas de Partição em Grafos (LAGUNA *et. al.*, 1994), Problema de Steiner (MARTINS *et. al.*, 1999a, MARTINS *et. al.*, 1999b), Problema do Clique Máximo em Grafos (PARDALOS *et. al.*, 1995).

GRASP também pode ser facilmente implementado em paralelo, dividindo as iterações entre diversos processadores. Em PARDALOS *et al.* (1996), os autores apresentam um GRASP paralelo para o problema MAX-SAT. Já em FEO *et. al.* (1995), um GRASP paralelo foi aplicado ao problema do clique máximo. Finalmente, em FEO

et. al. (1994), os autores apresentam um GRASP para resolver o Problema do Conjunto Independente em Grafos e uma versão paralela do método.

4.1.1. Princípios Básicos

Seja P um problema de otimização combinatória a ser resolvido aplicando a metaheurística GRASP. O pseudocódigo abaixo descreve um algoritmo básico para o GRASP:

```
Algoritmo GRASP()
Inicio
(1) DadosEntrada(P);
(2) Inicialização da melhor solução encontrada: S*;
(3) Enquanto critério de parada não satisfeito faça
(4)         Fase de Construção;
(5)         Fase de Busca Local;
(6)         Atualização de S*;
(7) Fim_enquanto;
```

Figura 4.1 – Algoritmo Básico GRASP

As linhas (1) e (2), da Figura 4.1, correspondem a inicialização dos dados de entrada do problema e da variável que representa a melhor solução encontrada, respectivamente. Ainda na Figura 4.1, linhas (3)-(7), estão descritas as iterações do GRASP que serão executadas até que o critério de parada definido seja atendido. Critérios de parada comumente utilizados em GRASP são: definição de um número máximo de iterações ou quando for encontrada uma solução desejada. Na linha (4), da Figura 4.1, está representada a fase de construção do método, onde, a cada iteração, uma solução inicial é construída. Já a linha (5), Figura 4.1, corresponde à fase de busca local, na qual tenta-se obter um ótimo local a partir de uma solução obtida na fase de construção. A cada iteração GRASP, S* é atualizada, como descrito na Figura 4.1, linha (6).

4.1.2. Fase de Construção

Na fase de construção, uma solução viável (S), para o problema P, é construída também iterativamente, elemento por elemento. A cada iteração, os elementos que podem ser inseridos em S são avaliados por uma função gulosa definida de acordo com o problema P, segundo o benefício alcançado pela inclusão do elemento na solução S. Os elementos avaliados são incluídos em uma lista, denominada Lista C, que é ordenada de forma crescente em relação ao benefício dos elementos na solução. Uma lista, denominada Lista Restrita de Candidatos (RCL – Restricted Candidate List) ou Lista de Melhores Candidatos, é formada com os elementos de melhor avaliação. O componente probabilístico de GRASP está na escolha aleatória de um elemento da RCL para ser inserido em S. O elemento escolhido pode, ou não, ser o de melhor avaliação. Além disso, a cada iteração, a escolha do próximo elemento é realizada considerando a escolha anterior, este é o fator adaptativo de GRASP. Na Figura 4.2 está descrito o pseudocódigo para a fase de construção.

```
Algoritmo FaseConstruçãoGRASP(S,g(C))
Inicio
(1)  S = Ø;
(2)  Lista de candidatos (C) ordenada de acordo com a função gulosa g: C → R;
(3)  enquanto C ≠ Ø  faça
(4)          Elabore RCL (RCL);
(5)          Selecione aleatoriamente um elemento de RCL;
(6)          Inclua o elemento em S;
(7)          Atualize a Lista C;
(8)  fim_enquanto;
Fim
```

Figura 4.2 – Pseudocódigo para Fase de Construção do GRASP

Na linha (1) da Figura 4.2, a solução S é inicializada. Enquanto que, na linha (2) da mesma figura, a Lista C é inicializada ordenando, de forma crescente, os elementos candidatos a serem inseridos em S de acordo com a função gulosa g: C → R. As

iterações, descritas nas linhas (3)–(8) da Figura 4.2, são realizadas até que uma solução viável seja construída. Ainda na Figura 4.2, linha 4, a lista restrita de candidatos, RCL, é elaborada. Na linha (5), um elemento da lista RCL é escolhido aleatoriamente e adicionado à solução S na linha (6) da Figura 4.2. Na linha (7), a lista C é atualizada. A primeira fase termina quando todos os elementos forem incluídos em S, ou seja, quando a Lista C estiver vazia.

Um fator importante em GRASP é o tamanho da Lista Restrita de Candidatos (RCL). A diversidade das soluções encontradas depende do tamanho da lista. Se |RCL|=1, a cada iteração da fase de construção, a escolha do elemento que vai compor S é uma escolha gulosa e as soluções, obtidas na fase de construção, serão iguais em todas as iterações do GRASP. Por outro lado, quanto maior o tamanho da lista RCL, maior a diversidade das soluções tornando mais claro o caráter aleatório da fase de construção.

A restrição dos elementos na RCL pode ser feita determinando um numero máximo de elementos na lista e/ou definindo um intervalo, no qual os elementos devem pertencer para que possam ser incluídos na RCL.

Um outro fator importante a ser discutido em relação à definição da lista RCL é a qualidade da solução obtida que depende da qualidade dos elementos que compõem a RCL e do tamanho da lista. Em relação à qualidade dos elementos da RCL, o critério guloso deve ser elaborado de forma cuidadosa, pois os elementos que irão compor a RCL são definidos de acordo com esse critério. Portanto, caso a função gulosa não seja bem definida, podem ser geradas soluções ruins na fase de construção. Além disso, o tamanho da RCL deve ser limitado para impedir que sejam escolhidos, a cada iteração da fase de construção, elementos que determinam um pequeno benefício à solução.

4.1.3. Fase de Busca Local

Antes de descrever esta fase do método GRASP, é preciso definir o conceito de vizinhança. Uma vizinhança N(S), para o problema P, é o elemento que introduz a noção de proximidade entre uma solução S e as demais soluções de P. Uma solução S é um ótimo local se não existe nenhuma solução melhor que S em N(S).

A solução gerada na fase de construção do GRASP não é, necessariamente, um ótimo local em relação a uma determinada vizinhança. Deste modo, é conveniente aplicar um método de busca local para tentar melhorar cada solução obtida na fase de construção. Assim, o dado de entrada para a fase de busca local do GRASP corresponde à solução inicial S obtida na fase de construção. Em cada iteração da fase de busca local, são realizadas tentativas de melhorias sucessivas na solução S através de uma busca na sua vizinhança. A segunda fase termina quando não é possível obter melhoras na solução S em N(S).

Dado uma vizinhança N, para uma solução S, o pseudocódigo apresentado na Figura 4.3 descreve a fase de busca local:

```
Algoritmo_FaseBuscaLocal(N(S),S)
Inicio
(1)  enquanto critério de parada não satisfeito faca
(2)          Encontre solução S' ∈ N(S) ;
(3)          Se S' for melhor que S então S =S';
(4)  fim_enquanto;
Fim:
```

Figura 4.3 – Pseudocódigo para Fase de Busca Local do GRASP

As linhas (1)-(4), da Figura 4.3, correspondem às iterações da fase de busca local que são executadas até que o critério de parada, definido para a fase de busca local, não seja atendido.

A eficiência do procedimento de busca local depende da definição correta da vizinhança, da estratégia de busca escolhida e da qualidade da solução inicial. Além disso, a utilização de estruturas de dados eficientes pode também contribuir para eficiência do procedimento de busca acelerando sua execução.

Uma das vantagens na utilização do GRASP se refere à qualidade da solução inicial, pois, caso os componentes da fase de construção sejam bem definidos, as soluções inicias geradas, pela própria definição do procedimento de construção, são consideradas de boa qualidade. Outro benefício intrínseco nesta metaheurística é a facilidade de implementação, sendo necessária a definição de poucos parâmetros como, por exemplo, a restritividade da lista de candidatos e o número de iterações. Além disso, normalmente, heurísticas gulosas são simples de se projetar e implementar.

Outra vantagem encontrada em GRASP é a variedade das soluções obtidas quando a realização de várias iterações é permitida. Para os problemas de otimização combinatória que são considerados difíceis, muitas vezes, as soluções ótimas não são conhecidas e, sendo assim, a crítica para distinguir entre soluções ótimas e próximas do ótimo não existe. Deste modo, a flexibilidade alcançada pela rapidez com que GRASP gera um grande número soluções pode ser bastante satisfatória quando da avaliação destas soluções.

4.2. VNS - Variable Neighborhood Search

Um algoritmo de busca local, quando aplicado em um problema de otimização, começa com uma solução viável do problema e a cada iteração tentar obter, na vizinhança da solução, um ótimo local. Portanto, para essa classe de algoritmos é extremamente importante definir corretamente a estrutura de vizinhança que será adotada. De maneira geral, quanto maior o tamanho vizinhança, melhor será a qualidade do ótimo local encontrado. Da mesma forma, quanto maior o tamanho da vizinhança, maior o tempo gasto pelo algoritmo de busca em cada iteração. Por outro lado, existe um risco de se obter soluções ruins caso a estrutura da vizinhança for definida de maneira que um espaço muito pequeno seja considerado. Muitos estudos têm sido realizados no sentido de desenvolver procedimentos heurísticos ou metaheurísticos, baseados nos métodos de busca local tradicionais, que diminuam o risco de se obter soluções ruins e que, por outro lado, sejam eficientes em relação ao tempo computacional exigido quando da sua aplicação.

Variable Neighborhoods Search (VNS) é uma metaheurística proposta por MANDENOVIC et al. (1997) baseada na exploração de uma estrutura de vizinhança dinâmica. No VNS o tamanho da vizinhança não é constante. Seja p o tamanho da vizinhança definido no método de busca escolhido para solucionar um problema, o VNS começa com um valor pequeno de p que cresce gradativamente até um valor máximo definido para o problema. Portanto, ao contrário dos outros métodos de busca local, VNS não segue uma trajetória, mas explora vizinhanças cada vez mais distantes da solução corrente em busca de um ótimo local de boa qualidade para o problema.

4.2.1. Princípios Básicos

Para descrever o VNS, considere P como um problema de otimização combinatória e S, uma solução viável de P. Além disso, $N_k(S)$ = {conjunto de soluções na k-ésima vizinhança da solução S} com k = 1, ..., k_{max}. A definição da estrutura de vizinhança adotada e o valor de k_{max} dependem do problema P. Nos métodos de busca local tradicionais, k_{max} = 1. Na Figura 4.4 está representado um pseudocódigo para o VNS.

```
Algoritmo_Básico_VNS
Inicio
(1)   DadosEntrada(P,S);
(2)   Defina a estrutura de vizinhança que será considerada pelo algoritmo de
      busca e o valor de k_max
(3)   k = 1;
(4)   enquanto k ≤ k_max faça
(5)         Escolha uma solução S' aleatoriamente na k-ésima vizinhança de
            S, ou seja, S' ∈ N_k(S);
(6)         Aplique um método de busca local com S' como solução inicial;
            Seja S", o ótimo local encontrado;
(7)         Se S" é melhor que S
            então faça k=1; S = S";
            senão faça k = k+1;
(8)   fim-enquanto;
Fim;
```

Figura 4.4 – Algoritmo básico para oVNS

Na linha (1) da Figura 4.4, é elaborada a solução inicial (S) para o problema P. A estrutura de vizinhança e o valor dos parâmetros k_{max} são definidos na linha (2) da Figura 4.4. A variável k, que controla o espaço de busca considerado pelo algoritmo em cada iteração, é inicializada na linha (3) da mesma figura. Já nas linhas (4)-(8), estão descritas as iterações do VNS. Na linha (5) da Figura 4.4, uma solução S' pertencente a

k-ésima vizinhança de S é escolhida aleatoriamente. Ainda na Figura 4.4, linha (6), é encontrado um ótimo local na vizinhança de S' através da aplicação do algoritmo de busca definido. O ótimo local é denominado S". Na linha (7) da Figura 4.4, se S" for melhor que S, a busca recomeça com k = 1, considerando a solução S", caso contrário, a busca continua em S com k=k+1.

A escolha do método de construção da solução inicial considerada pelo algoritmo é extremamente importante na elaboração do VNS, pois dela depende a qualidade da solução inicial. Além disso, podem ocorrer ciclos caso não seja estabelecido algum critério adicional quando é realizada a escolha aleatória de uma solução na vizinhança da solução que está sendo considerada (linha (5) da Figura 4.4).

Em relação à definição do espaço de busca considerado, é importante destacar que o tempo computacional exigido pela execução de um algoritmo de busca é diretamente proporcional ao tamanho da vizinhança considerada. Por outro lado, quanto maior o tamanho da vizinhança, menores serão as chances de se encontrar ótimos locais pobres. Nas heurísticas de busca local tradicionais, o tamanho do espaço de busca é fixo durante toda a execução do método. Por outro lado, no VNS, a partir de uma análise bem elaborada do problema, é possível definir uma variação desse espaço de busca de maneira que se tenha soluções melhores sem provocar um grande aumento no tempo computacional exigido.

5. Heurísticas para o Problema do Caixeiro Viajante com Grupamentos

Neste capítulo, serão descritas as heurísticas que foram usadas para solucionar o Problema do Caixeiro Viajante com Grupamento. O método de solução do PCVG, inicialmente, é baseado na idéia de CHISMAN (1975), ou seja, o PCVG é transformado em um PCV e resolvido aplicando um dos métodos de solução disponíveis para o PCV.

Outra idéia proposta foi perturbar os dados de entrada gerando variações do problema e analisar o comportamento dos algoritmos quando executados considerando essas variações. Para isso, a cada instância do PCVG, foram geradas um número α de variações. Essas variações foram solucionadas usando as heurísticas implementadas. Por fim, as soluções obtidas para as variações eram modificadas obtendo soluções para o problema original.

A seguir será descrita a maneira como as variações foram geradas e como os algoritmos foram elaborados. No decorrer deste trabalho, considere:

- $G = (V, E)$: grafo simétrico e completo
- V: {conjunto de vértices de G} = $\{v_1, \ldots, v_i, \ldots, v_j, \ldots, v_n\}$, $|V| = N$ e
- E: {conjunto de arestas de G}
- $PCVG_G$: Instância do Problema do Caixeiro Viajante com Grupamento definido no grafo G
- PCV_G: Instância do Problema do Caixeiro Viajante definida a partir do $PCVG_G$
- $V=V_1 \cup \ldots \cup V_I \cup V_J \cup \ldots \cup V_M$, onde M é o número de grupamento do $PCVG_G$
- d_{ij}: custo da aresta $(v_i,v_j) \in E$ com $\{v_i,v_j\} \in V$
- S: ciclo hamiltoniano, solução do $PCVG_G$
- C_S: custo da solução S

- $PCVG_I$: i-esima variação do $PCVG_G$
- d^I_{ij}: custo da aresta (v_i,v_j) na variação $PCVG_I$

Considere ainda os seguintes conceitos:

- Um vértice v_j *é mais próximo de* v_i, se (v_i,v_j) é a aresta de menor custo que possui v_i como vértice incidente.
- Um vértice v_i *é vizinho de* v_j *em S*, se existe a aresta (v_i,v_j) em S

Na Figura 5.1, está representado o pseudocódigo do algoritmo implementado para gerar as variações. O algoritmo foi denominado Algoritmo Gerador.

Algoritmo Gerador (PCVG$_G$, PCVG$_I$)
Inicio
(1)Defina β *como* $\beta << máx\{d_{ij}\}$; $i = 0$;
(2)Enquanto $i < \alpha$ *faça*
(3) Para cada aresta (v_i,v_j) *do grafo, tal que,* v_i *e* v_j *pertencem ao mesmo grupamento faça:*
(4) Defina d escolhendo, aleatoriamente, um dos valores
$0, \beta, 2\beta, 3\beta, -\beta, -2\beta, -3\beta$
(5) Some d ao custo da aresta (v_i,v_j) *tomando o cuidado para que o novo valor do custo seja positivo*
$d^I_{ij} = d + d_{ij}$;
(6) Fim Para;
(7)Fim Enquanto;
Fim

Figura 5.1 : Algoritmo Gerador

A instância original do PCVG definida no grafo G, $PCVG_G$, é o dado de entrada do algoritmo. Enquanto que, as α variações do problema são a saída do algoritmo. Na i-ésima iteração do loop, descrito nas linhas (2)-(4) da Figura 5.1, a i-ésima variação é obtida alterando os custos das arestas do $PCVG_G$ que conectam vértices que pertencem

ao mesmo grupamento. Para isto, a cada aresta (v_i,v_j), se v_i e v_j pertencem ao mesmo grupamento então, na linha (3) da Figura 5.1, é definido o número *d* que é somado ao custo da aresta d_{ij} na linha (4), resultando em um novo custo d^I_{ij}.

Depois de geradas as variações, a instância original e suas variações foram solucionadas usando os algoritmos elaborados segundo a idéia de CHISMAN (1975).

Os cinco algoritmos implementados diferem-se pela heurística usada para solucionar a instância do PCV gerada a partir da instância original, PCV_G. Em três algoritmos, o PCV_G é resolvido aplicando heurísticas de construção elaboradas para o PCV que foram propostas por KINDERVATER *et al.* (1989). No primeiro, denominado HIMP, o PCV_G é resolvido usando a **H**eurística da **I**nserção **M**ais **P**róxima. No método denominado HIMB, a heurística usada foi a **H**eurística da **I**nserção **M**ais **B**arata. No terceiro método, HVMP, o PCV_G é resolvido com a **H**eurística do **V**izinho **M**ais **P**róximo.

Baseado em H1, método proposto por GENDREAU *et al.* (1996a), foi desenvolvido um método, denominado H1M, onde o PCV_G foi solucionado por GENIUS-M que é uma heurística proposta neste trabalho, desenvolvida a partir de GENIUS.

Finalmente, para comparar com algoritmos propostos, H1 foi implementado. Como dito anteriormente, no H1, o PCV_G é solucionado usando a heurística GENIUS.

A seguir, na Figura 5.2, é apresentado o pseudocódigo dos algoritmos. Os dados de entrada dos algoritmos podem ser a instância original do PCVG ou uma de suas variações. Na linha (1) da Figura 5.2, a penalidade L é definida para criar, na linha (2) da mesma figura, a instância do PCV baseada na instância original do PCVG, denominada PCV_G. Na linha (3) da Figura 5.2, o PCV_G é solucionado. O que diferencia os algoritmos implementados é o método usado para solucionar o PCV_G. Ainda na Figura 5.2, linha (4), a solução real do problema é obtida através de um procedimento de "concerto" da solução do PCV_G, S_G, para que esta seja uma solução real do PCVG.

Na linha (5), se o problema resolvido for o $PCVG_G$, a solução obtida, na linha (4), é a solução real do problema e o algoritmo termina. Caso contrário, se o problema resolvido for uma variação da instância original (perturbação dos dados de entrada), a solução S é solução da variação que está sendo considerada. Desta forma, na linha (7) da Figura 5.2, os custos das arestas que fazem parte de S e que foram modificados na geração da variação deverão ser alterados para o valor original, obtendo uma solução válida para a instância original do PCVG.

Algoritmo Heurístico (Instância do PCVG)
Início
(1) Defina $L \in N$ *como* $L >> max\{d_{ij}\}$.
(2) Crie uma instância do PCV somando L ao custo de toda aresta (v_i, v_j) *do* $PCVG_G$ *onde* v_i *e* v_j *pertençam a grupamentos diferentes. Seja* PCV_G, *a instância do PCV criada com as modificações.*
(3) Resolva o PCV_G;
Seja S_G, *a solução obtida e* C_G *o seu custo;*
(4) Diminua L dos custos de toda aresta de S_G *cujos vértices pertençam a grupamentos diferente, obtendo a solução real S que é a solução do problema considerado:*
C *(custo da solução S)* $= C_G - L*M$
(5) Se a instância considerada for a original
então pare, S é a solução do problema
(6) senão
Seja $PCVG_I$ *a variação que está sendo considerada.*
(7) Para cada aresta (v_s, v_t) *da solução S que conecta vértices do mesmo grupamento faça:*
$C = C - d^l_{st} + d_{st}$
(8) Fim-Para;
Fim

Figura 5.2- Algoritmo Heurístico

Desta forma, nos procedimentos aqui propostos, foram realizadas dois tipos de perturbação nos dados de entrada do PCVG: P1 e P2.

- P1: Na perturbação P1, todos os custos das aresta (v_i,v_j), tal que os vértices v_i e v_j pertencem a grupamentos distintos, devem ser alterados. Com isso, o PCVG é transformado em um PCV.
- P2: A segunda perturbação dos dados de entrada, P2, se refere ao passos 4 e 5 do algoritmo mostrado na Figura 5.1. Neste caso, a idéia é obter novas soluções aproximadas e de boa qualidade sem modificar o algoritmo, mas perturbando os dados de entrada. Com isso, toda heurística ou metaheurística pode, a cada perturbação, gerar diferentes soluções aproximadas.

A seguir serão descritas as heurísticas implementadas para resolver a instância do PCV associada à instância do PCVG em cada um dos métodos.

5.1. Heurística da Inserção Mais Próxima

A heurística de construção, denominada Heurística da Inserção mais Próxima, foi usada no algoritmo HIMP para resolver a instância do PCV resultante do PCVG. Na Figura 5.3 é descrito o pseudocódigo desta heurística.

Inicio
(1) Selecione aleatoriamente um vértice $v_0 \in V$;
(2) $S = v_0$;
(3) $C_S = 0$;
(4) Enquanto S não contém todos os vértices de V faça
(5) Selecione o vértice v_s *a ser inserido no ciclo como um vértice, não pertencente a S, mais próximo a um dos vértices que já pertencem a S.*
(6) Para cada par de vértices $\{v_i, v_j\}$ *vizinhos em S defina*

$$C_S(i,j) = C_S - d_{ij} + d_{is} + d_{sj}$$

(7) Escolha v'_i *e* v'_j, *tal que, em (6),* $C_S(i',j')$ *seja o mínimo possível;*
(8) Insira v_s *entre* v'_i *e* v'_j, *ou seja,* $S = S \cup \{v_s\}$;
(9) $C_S = C_S(i',j')$;
(10) Fim-Enquanto;
Fim

Figura 5.3- Heurística da Inserção mais Próxima

Na linha (2) da Figura 5.3, a solução S é inicializada com o vértice $v_0 \in V$ que escolhido aleatoriamente na linha (1) da mesma figura. O loop, descrito nas linhas (4)-(10) da Figura 5.3, é realizado enquanto existir um vértice $v_i \in V$ que não pertença à S. Na linha (5) da Figura 5.3, o próximo vértice a ser inserido em S, v_s, é escolhido tomando um vértice não pertencente à S, mais próximo de um dos vértices que já pertence à S. A posição onde v_s será incluído em S é definida nas linhas (6) e (7) da Figura 5.3. Nas linhas (8) e (9), a solução S e seu custo são atualizados.

5.2. Heurística da Inserção Mais Barata

No algoritmo denominado HIMB, a heurística de construção usada para resolver a instância do PCV associada à instância original do PCVG foi a Heurística da Inserção mais Barata. O pseudocódigo para a heurística é descrito na Figura 5.4.

Inicio
(1) Selecione aleatoriamente um vértice $v_0 \in V$;
(2) $S = v_0$;
(3) $C_S = 0$;
(4) Enquanto S não contém todos os vértices de V faça
(5) Para todo vértice v_s *que não pertence à S e todo par de vértices* $\{v_i, v_j\}$ *vizinhos em S, faça*
$C_S(i,s,j) = C_S + d_{is} + d_{sj} - d_{ij}$;
(6) Fim-Para;
(7) Selecione os vértice v'_s, v'_i *e* v'_j, *tal que, em (5),* $C_S(i',s',j')$ *seja mínima possível;*
(8) Insira v'_s *em S entre os vértices* v'_i *e* v'_j;
(9) $C_S = C_S(i',s',j')$;
(10) Fim_Enquanto;
Fim:

Figura 5.4- Heurística da Inserção mais Barata

Com o vértice $v_0 \in V$ que foi escolhido aleatoriamente na linha (1) da Figura 5.4, a solução S é inicializada na linha (2) da mesma figura. O loop, descrito nas linhas (4)-(10) da Figura 5.4, é realizado enquanto existir um vértice $v_i \in V$ que não pertence à S. Na linha (7) da mesma figura, o próximo vértice, v_s, e a posição onde v_s será inserido em S são escolhidas segundo o valor definido nas linhas (5) e (6). Finalmente, nas linhas (8) e (9) da Figura 5.4, a solução S e seu custo são atualizados.

5.3. Heurística do Vizinho Mais Próximo

A próxima heurística descrita é a Heurística do Vizinho mais Próximo que foi usada no algoritmo HVMP para resolver a instância do PCV associada ao PCVG. A Figura 5.5 descreve o pseudocódigo para a heurística.

Inicio
(1) Selecione aleatoriamente um vértice $v_0 \in V$;
(2) $S = v_0$;
(3) $C_S = 0$;
(4) v_u = *vértice inserido em S mais recentemente* = v_0;
(5) Enquanto S não contém todos os vértices de V faça
(6) Escolha o vértice v_s *não pertencente à S mais próximo de* v_u *(vértice inserido em S mais recentemente).*
(7) Insira v_s *depois de* v_u ;
(8) $C_S = C_S + d_{us}$;
(9) $v_u = v_s$;
(10) Fim-Enquanto;
Fim;

Figura 5.5- Heurística do Vizinho Mais Próximo

Assim como nas heurísticas descritas anteriormente, o ciclo hamiltoniano, S, também é construído começando de um vértice escolhido aleatoriamente (linhas 1 e 2 da Figura 5.5). No loop, descrito nas linhas (5)-(10) da Figura 5.5, o próximo vértice inserido em S é escolhido, na linha (6), tomando o vértice mais próximo do vértice que foi inserido em S na iteração anterior. Nas linhas (7),(8) e (9) da mesma figura, a solução S, o custo de S e a variável que representa o vértice inserido em S mais recentemente são atualizadas.

5.4. Heurística GENIUS

GENIUS é usado no método proposto por GENDREAU et al. (1996a), denominado H1, para resolver o PCV resultante do PCVG penalizando as arestas que conectam vértices de grupamentos distintos.

GENIUS é uma heurística proposta, inicialmente, por GENDREAU *et al.* (1992) para solucionar o Problema do Caixeiro Viajante. Comparada às outras heurísticas convencionais disponíveis na literatura do PCV, GENIUS é considerada, atualmente, uma das melhores. Como mencionado anteriormente, o método consiste de uma fase de construção, denominada GENI (Generalized Insertion Procedure), seguida por uma fase de otimização, denominada US (Unstringing e Stringing).

Na fase de construção, GENI, é obtida uma solução viável (ciclo hamiltoniano) para o PCV. A solução é gerada começando por um ciclo contendo três vértices escolhidos aleatoriamente e inserindo, a cada passo, um vértice que ainda não pertence à solução.

Suponha que o vértice v foi escolhido para ser inserido entre dois vértices v_i e v_j do ciclo parcial. Uma das características principais deste método de construção é que, antes da inserção de v, os vértices v_i e v_j não são, necessariamente, vizinhos no ciclo. Porém, depois de inserido o vértice v, os dois vértices, v_i e v_j, se tornam adjacentes à v.

A seguir, será dada a notação adotada para descrever o método. Para uma dada orientação do ciclo, considere:

- P_{ij}: conjunto de vértices do caminho entre os vértices v_i e v_j
- v_k: um vértice do caminho entre v_j e v_i
- v_l: um vértice do caminho entre v_i e v_j
- $N_p(v)$ = {conjunto de p vértices mais próximos de v, pertencentes ao ciclo}

Além disso, para todo vértice v_h pertencente ao ciclo

- v_{h+1} é o sucessor do vértice v_h no ciclo
- v_{h-1} é vértice antecessor de v_h no ciclo

Suponha que S é a solução que está sendo construída. A cada iteração do GENI, o vértice que será inserido em S é escolhido aleatoriamente entre os vértices que ainda não pertencem à S. Seja v, o vértice escolhido. Existem dois tipos de inserção de vértices na fase GENI: Tipo 1 e Tipo2

Tipo 1: Escolha v_r, $v_s \in N_p(v)$ e $v_k \in \{N_p(v_{r+1}) \cap P_{sr} \backslash \{v_r, v_s\}\}$. Eliminar as arestas (v_r, v_{r+1}), (v_s, v_{s+1}) e (v_k, v_{k+1}) e inserir (v_r, v), (v, v_s) e (v_{r+1}, v_k) e (v_{s+1}, v_{k+1}) no ciclo. Observe que, a orientação dos caminhos $(v_{r+1}, \ldots, v_s)$ e $(v_{s+1}, \ldots, v_k)$ é invertida depois da inserção de v. A Figura 5.6 representa a inserção do Tipo 1.

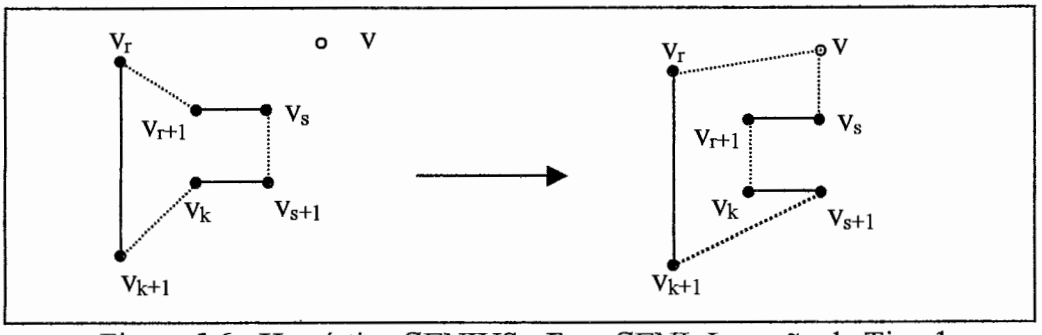

Figura 5.6 - Heurística GENIUS - Fase GENI: Inserção do Tipo 1

Tipo 2: Escolha v_r, $v_s \in N_p(v)$ e $v_k \in \{N_p(v_{r+1}) \cap P_{sr} \setminus \{v_s, v_{s+1}\}\}$ e $v_l \in \{N_p(v_{s+1}) \cap P_{rs} \setminus \{v_r, v_{r+1}\}$. Eliminar as arestas (v_r, v_{r+1}), (v_{l-1}, v_l), (v_s, v_{s+1}) e (v_{k-1}, v_k) e inserir (v_r, v), (v, v_s), (v_l, v_{s+1}), (v_{k-1}, v_{l-1}) e (v_{r+1}, v_k) no ciclo. A orientação dos caminhos $(v_{r+1}, \ldots, v_{l-1})$ e $(v_l, \ldots, v_s)$ é invertida depois da inserção de v. A Figura 5.7 ilustra esse tipo de inserção:

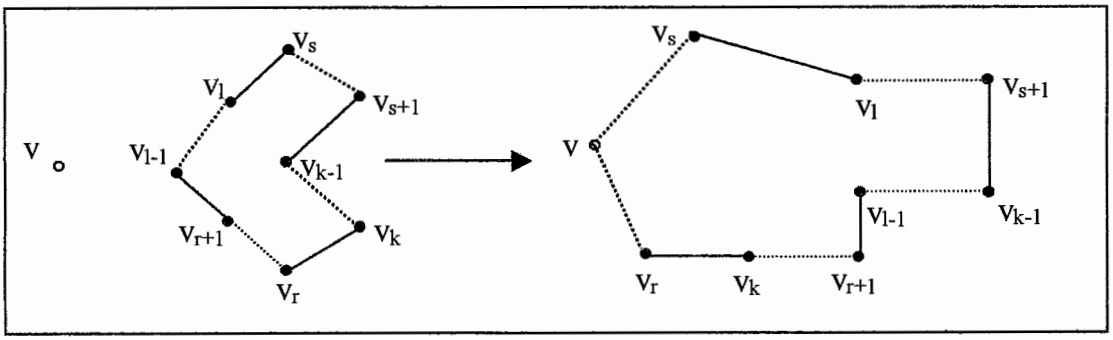

Figura 5.7 - Heurística GENIUS - Fase GENI: Inserção do Tipo 2

A melhor posição para inserir v no ciclo parcial é determinada comparando os dois tipos de inserção e considerando as duas possíveis orientações do grafo.

Nos dois tipos de inserção, o número de possíveis escolhas viáveis de v_r, v_s, v_k, v_l é da ordem de $p^4(O(p^4))$. Quando p é pequeno comparado ao número de nós do grafo (n), a complexidade do método diminui consideravelmente, pois caso não fosse definida a estrutura $N_p(v)$, o número de escolhas possíveis dos vértices v_r, v_s, v_k, v_l seria da ordem de n^4. Deste modo, a escolha de p interfere diretamente na qualidade da solução e no tempo de execução do algoritmo, pois determina o número de possíveis inserções que serão consideradas. Normalmente, p é um número pequeno (p=3 ou 4).

A complexidade de GENI pode ser definida como se segue:

- (1) Número de Operações para escolha dos vértices v_r, v_s, v_k, v_l : $O(p^4)$
- (2) Número de Operações para inserção de um vértice no ciclo: O(n)

Como o ciclo começa com 3 vértices escolhidos aleatoriamente, (1) e (2) são executadas (n-3) vezes, portanto a complexidade de GENI é da ordem de $O(np^4 + n^2)$ operações.

Terminado a fase GENI, a próxima fase de GENIUS é a de otimização local, denominada US, onde o objetivo é tentar melhorar a solução construída na fase anterior removendo um vértice do ciclo através da operação reversa de GENI e inserindo-o de volta usando GENI. O procedimento termina quando nenhuma melhora pode ser obtida removendo e reinserindo um vértice qualquer da solução.

Nesta fase são consideradas duas maneiras de retirada de um vértice do ciclo. Suponha que v_r é vértice que será retirado do ciclo:

Retirada Tipo 1: Escolha $v_s \in N_p(v_{r+1})$. Para uma dada orientação de G, escolha $v_k \in N_p(v_{r-1})$ como um vértice do caminho $(v_{r+1}, ..., v_{s-1})$. As arestas (v_{r-1},v_r), (v_r,v_{r+1}), (v_k,v_{k+1}) e (v_s,v_{s+1}) são removidas, enquanto que as arestas (v_{r-1},v_k), (v_{r+1},v_s) e (v_{k+1},v_{s+1}) são inseridas no ciclo. A orientação dos caminhos $(v_{r+1}, ..., v_k)$ e $(v_{k+1}, ..., v_s)$ são invertidas. A retirada do Tipo 1 é ilustrada na Figura 5.8.

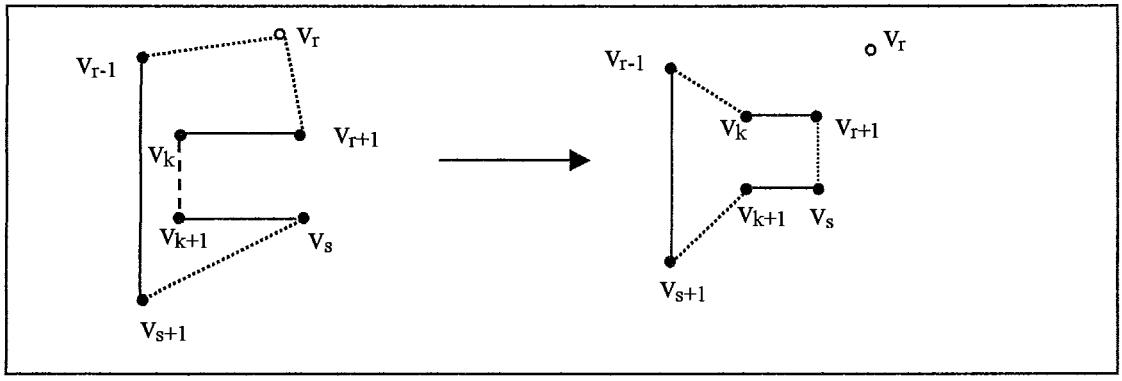

Figura 5.8 - Heurística GENIUS - Fase US: Retirada Tipo 1

Depois de retirado do ciclo, o vértice v_r é reinserido usando o GENI.

Retirada Tipo 2: Escolha $v_s \in N_p(v_{r+1})$. Para uma dada orientação de G, escolha $v_k \in N_p(v_{r-1})$, como um vértice do caminho $(v_{s+1}, ..., v_{r-2})$, e $v_l \in N_p(v_{k+1})$, como um vértice do caminho $(v_s, ..., v_{k-1})$;. As arestas (v_{r-1},v_r), (v_r,v_{r+1}), (v_{s-1},v_s) e (v_l,v_{l+1}) e (v_k,v_{k+1}) são removidas, enquanto que as arestas (v_{r-1},v_k), (v_{l+1},v_{s-1}), (v_{r+1},v_s) e (v_l,v_{k+1}) são inseridas no ciclo. A orientação dos caminhos $(v_{r+1}, ..., v_{s-1})$ e $(v_{l+1}, ..., v_k)$ são invertidas. A Figura 5.9 ilustra a retirada do tipo 2.

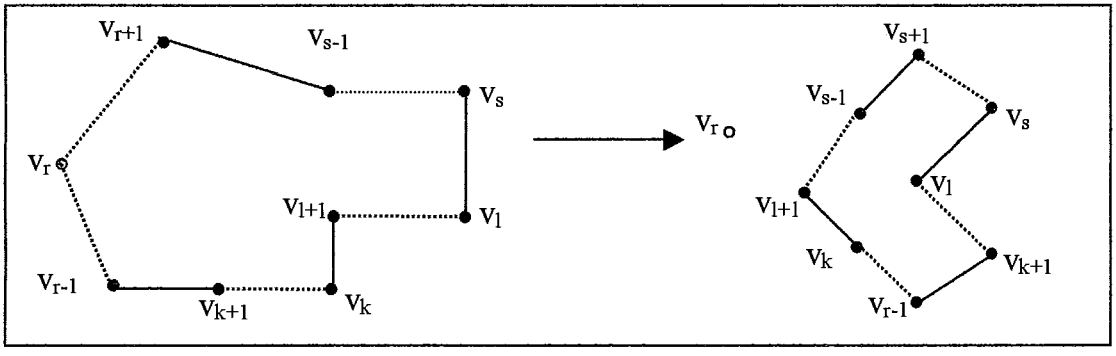

Figura 5.9 - Heurística GENIUS - Fase US: Retirada Tipo 2

Depois de retirado do ciclo, o vértice v_r é reinserido usando o GENI.

Na fase US, são considerados os dois tipos de retirada e as duas possíveis orientações do grafo. Na Figura 5.10, é apresentado o pseudocódigo para a segunda fase do GENIUS:

```
Algoritmo US
1- Considere um ciclo inicial S com custo C. Considere S* como a melhor solução
encontrada até o momento e C* o seu custo, então S*=S, C*=C e t=1;
2- Retire o vértice v_t de S e tente reinseri-lo considerando os dois tipos de retirada, os dois
tipos de inserção e as duas possíveis orientações de S.
Seja S' o ciclo obtido e C', o seu custo. Faça S=S' e C=C';
3- Se C < C*
    então S*= S, C*=C, t = 1, volte para 2;
    senão
        t=t +1;
        Se t=n+1
            então pare, melhor solução é S* e seu custo é C*;
            senão volte para 2;
        Fim-se;
Fim-se;
Fim Algoritmo US;
```

Figura 5.10 - Pseudocódigo US

O ciclo inicial S é construído por GENI. Na linha (2) da Figura 5.10, a cada iteração da fase US, um vértice de S é retirado e reinserido considerando as duas possíveis orientações de S, os dois tipos de retirada definidos no US e os dois tipos de inserção definidos no GENI. Na linha (3) da mesma figura, se a solução encontrada com a retirada e reinserção do vértice v_i (S') for melhor que S*, então a busca recomeça considerando S' e o vértice v_1. Por outro lado, se o custo de S* é menor que o de S', a busca continua tentando reinserir o vértice v_{i+1} em S.

Observe que tanto GENI quanto US podem ser usados separadamente em outros métodos. Este fato e por GENI + US ser considerada uma das melhores heurísticas

existentes atualmente para o PCV, motivou a análise de variantes desta técnica neste trabalho.

5.5. Heurística GENIUS-M

No algoritmo H1M, GENIUS-M foi usada para resolver o PCV resultante da instância original do PCVG. A Heurística GENIUS-M foi elaborada baseada em GENIUS. A fase de construção de GENIUS-M é idêntica a de GENIUS. A diferença entre as duas heurísticas está apenas na fase de otimização que, em GENIUS-M, foi denominada US-M.

A Figura abaixo descreve o pseudocódigo de US-M:

Algoritmo US-M

1- Considere um ciclo inicial S com custo C. Seja S, a melhor solução encontrada até o momento e C* o seu custo. Faça S*= S e C*=C e t=1;*

2- Retire o vértice v_t de S e tente reinseri-lo considerando os dois tipos de retirada e os dois tipos de inserção definidos em GENIUS e as duas possíveis orientações de S.
Seja S' o ciclo obtido e C', o seu custo. Faça S=S' e C=C';

*3- Se C < C**
então S= S; C*=C;*

4- Faça t =t +1;

5- Se t= n+1
então pare, melhor solução é S e seu custo é C*;*
senão volte para 2;

6-Fim-se;

Fim Algoritmo US-M;

Figura 5.11 - Pseudocódigo do US-M

No GENIUS-M, assim como em GENIUS, a cada vértice retirado e reinserido no ciclo, são considerados os dois tipos de retirada, os dois tipos de inserção e as duas orientações do grafo (linha 2 da Figura 5.11). Em US, toda vez que a retirada e reinserção de um vértice em S resulta em uma solução S' melhor que S*, o algoritmo

passa analisar S' começando do primeiro vértice. Já em US-M, se v_i é o vértice que, retirado e reinserido em S, originou um ciclo S' melhor que S*, S' passa ser analisada pelo algoritmo a partir do vértice v_{i+1} (linha 3 da Figura 5.11).

No capítulo 7 deste trabalho, são apresentados os resultados obtidos com a aplicação das heurísticas propostas, HIMP, HIMB, HVMP e H1M, comparadas aos resultados obtidos com H1 (GENDREAU *et al.*,1996b). Além disso, para cada instância do PCVG, foram geradas 50 variações através de perturbações do tipo P2. As variações foram solucionadas pelas heurísticas descritas acima, gerando novas soluções aproximadas para a instância original do problema. Os resultados e comparações entre essas soluções são apresentados no capítulo 7.

6. Metaheurísticas para o Problema do Caixeiro Viajante com Grupamentos

Neste capítulo, serão descritos os algoritmos para o PCVG que foram desenvolvidos baseados nas metaheurísticas GRASP e VNS.

Foram elaborados quatro algoritmos. No primeiro, foi desenvolvido um algoritmo aplicando as idéias propostas no GRASP para resolver o PCVG sendo possível identificar todas as etapas descritas no GRASP padrão. No segundo algoritmo, algumas idéias do GRASP foram associadas à heurística GENIUS. Já o terceiro algoritmo elaborado combina a heurística GENIUS com as idéias propostas no VNS. Finalmente, o quarto algoritmo proposto foi elaborado baseado na heurística GENIUS associada às metaheurísticas GRASP e VNS.

Em três dos quatro algoritmos propostos, a idéia de CHISMAN (1975) foi usada para resolver o PCVG, ou seja, para resolver a instância do PCVG, é obtida uma instância do PCV através de modificações nos custos das arestas do grafo que definiu o PCVG (perturbação do tipo P1). A instância do PCV é solucionada pelos algoritmos propostos e a solução obtida é modificada para que se torne uma solução da instância original do PCVG. Em todos os três algoritmos, a instância do PCV é obtida somando L ao custo das arestas do PCVG que conectam vértices pertencentes a grupamentos diferentes, sendo que L é o valor máximo dos custos associados às arestas do PCVG.

Além disso, o procedimento descrito no capítulo anterior, onde são geradas variações da instância original do problema para obter novas soluções, também foi usado na elaboração de dois dos quatro algoritmos propostos.

No decorrer deste capítulo, os algoritmos são descritos.

6.1. GRASP padrão para o PCVG

O primeiro algoritmo desenvolvido resolve diretamente o PCVG, sendo que, os componentes básicos do GRASP foram usados para solucionar o problema. Portanto, o algoritmo consiste de duas fases, uma fase de construção e uma fase de busca local. A solução inicial do problema, denominada S_0, é obtida na fase de construção. Enquanto que, na fase de busca local, o objetivo é encontrar soluções melhores explorando a vizinhança da solução S_0.

Antes de começar a descrever o algoritmo, serão definidos alguns conceitos:

- Um grupamento *V_J é vizinho do grupamento V_I* se na solução do PCVG, V_J é visitado imediatamente depois de V_I
- A *distancia máxima* entre dois grupamentos distintos V_I e V_J, denominada D_{IJ}, é calculada da seguinte maneira:

  $D_{IJ} = \max\{d_{ij}\}$ tal que d_{ij} é o custo da aresta (v_i, v_j) com $v_i \in V_I$ e $v_j \in V_J$

6.1.1. Fase de Construção

Na fase de construção do algoritmo GRASP padrão, S_0 é obtida resolvendo, separadamente, os dois subproblemas definidos no PCVG: o Problema Inter-Grupamento (PIG) e o Problema Entre-Grupamento (PEG). Primeiro, o PEG é resolvido, e então, considerando a solução do PEG, o PIG é solucionado.

6.1.1.1. Fase de Construção: Solução do Problema Entre-Grupamentos (PEG)

No PEG, o objetivo é definir uma seqüência ótima entre os grupamentos do grafo começando de um grupamento V_0, escolhido aleatoriamente, visitando os demais grupamentos do grafo e retornando à V_0. A seqüência obtida, ou seja, a solução do PEG, é denominada S_{PEG}.

Suponha que V_I seja o grupamento inserido mais recentemente em S_{PEG}. Para determinar o grupamento vizinho de V_I é definida uma lista $C_1(V_I)$ que contém os grupamentos candidatos a serem vizinhos de V_I. A lista $C_1(V_I)$ é elaborada ordenando, de forma crescente, os grupamentos do PCVG diferentes de V_I e que não pertencem a S_{PEG}, segundo a distância máxima entre V_I e os demais grupamentos.

Os k_1 primeiros elementos de $C_1(V_I)$, ou seja, os k_1 grupamentos mais próximos de V_I, são escolhidos para compor a $RCL_1(V_I)$: lista restrita de candidatos (Restricted Candidate List) a vizinho de V_I. Sendo que k_1, tamanho da RCL_1, é um dado de entrada do algoritmo e escolhido de acordo com o problema.

O grupamento vizinho de V_I é determinado escolhendo, aleatoriamente, um grupamento V_J pertencente a $RCL_1(V_I)$.

O PEG é solucionado quando, para um determinado grupamento V_I, a lista de candidatos é vazia, significando que todos os demais grupamentos do grafo, diferentes de V_I, já pertencem à solução. Sendo assim, o grupamento que deve ser visitado depois de V_I deve ser o primeiro grupamento escolhido para compor S_{PEG}.

6.1.1.2. Fase de Construção: Solução do Problema Inter-Grupamentos (PIG)

No PIG, o objetivo é definir a seqüência ótima de visita entre os vértices do grafo. No GRASP padrão desenvolvido para o PCVG, a solução do PIG, denominada S_{PIG}, é obtida em duas etapas.

Na primeira etapa, para cada par de grupamentos vizinhos em S_{PEG}, V_I e V_J, são estabelecidos os vértices que serão a saída do grupamento V_I e a entrada do grupamento V_J. Na segunda etapa, para cada grupamento do PCVG, é definido um caminho que visita contiguamente todos os vértices do grupamento, considerando os vértices

escolhidos como entrada e saída em cada grupamento como vértice inicial e final de cada caminho.

Seja G = (V,E) um grafo completo e simétrico onde o PCVG pode ser definido. Na Figura 6.1, estão representadas as etapas do algoritmo GRASP padrão para o PCVG. Para facilitar a ilustração, no quadro 1 da Figura 6.1, somente os vértices do grafo foram representados. Ainda na Figura 6.1, quadro 2, está representada S_{PEG}, enquanto que, nos quadros 3 e 4 estão representadas as etapas para construção da S_{PIG}.

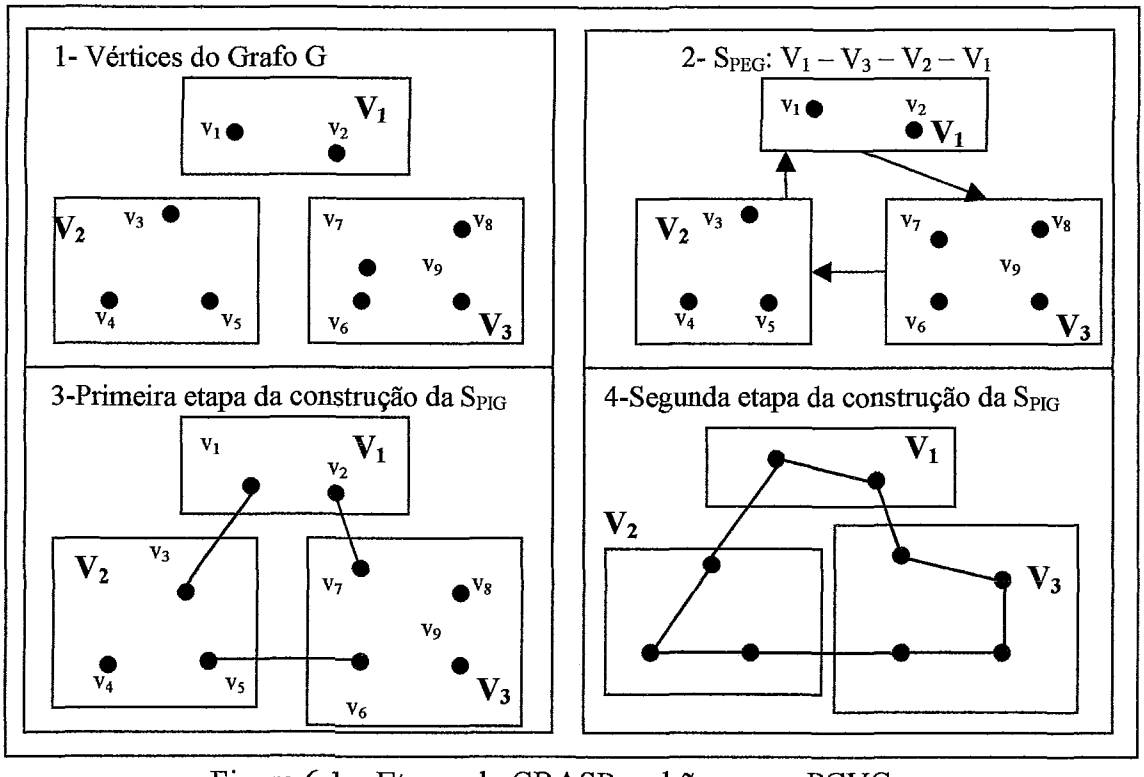

Figura 6.1 – Etapas do GRASP padrão para o PCVG

Na primeira etapa da construção da S_{PIG}, onde o objetivo é estabelecer os vértices que serão a entrada e a saída de cada grupamento, para cada par de grupamentos vizinhos em S_{PEG}, V_I e V_J, é definida uma lista, denominada $C_2(V_I,V_J)$, que contém as arestas entre os vértices v_i e v_j que são candidatos à saída do grupamento V_I e entrada do grupamento V_J, respectivamente. A lista $C_2(V_I,V_J)$ é elaborada ordenando as arestas de forma crescente segundo seus custos. Portanto:

- $C_2(V_I,V_J)$ = Lista de arestas $\{(v_i, v_j)$, com $v_i \in V_I$ e $v_j \in V_J\}$ ordenada pelo custo d_{ij}

Os k_2 primeiros elementos de $C_2(V_I,V_J)$, ou seja, as k_2 arestas de menor peso entre vértices v_i e v_j pertencentes aos grupamentos V_I e V_J, respectivamente, são escolhidas para compor a Lista Restrita de Candidatos à saída e entrada dos grupamentos V_I e V_J, denominada $RCL_2(V_I,V_J)$. Sendo que k_2 é um dado de entrada do algoritmo e definido de acordo com o problema.

- $RCL_2(V_I,V_J) = \{ k_2$ primeiras arestas $(v_i,v_j) \in C_2(V_I,V_J)\}$

Assim, uma aresta da RCL_2 é escolhida, aleatoriamente, definindo os vértices de saída e entrada nos grupamentos V_I e V_J, respectivamente.

Conhecida a seqüência de visitas entre os grupamentos e os vértices correspondentes aos vértices de entrada e saída em cada grupamento, resta definir a seqüência de visitas dos demais vértices pertencentes a cada grupamento. O problema então é encontrar, para cada grupamento V_I do PCVG, um caminho de custo mínimo, denominado P_I, que visita todos os vértices do grupamento considerando como vértices inicial e final aqueles que foram definidos no passo anterior como entrada e saída do grupamento. No algoritmo, o caminho foi construído através da Heurística da Inserção mais Próxima, descrita no capítulo anterior.

Feito isto, a fase de construção do GRASP padrão desenvolvido para o PCVG termina obtendo uma solução viável para o problema.

6.1.2. Fase de Busca Local

Na fase de busca local do algoritmo, é realizada uma busca em uma vizinhança da solução S_0, encontrada na fase anterior, com o objetivo de tentar encontrar soluções melhores que S_0.

Suponha que a seqüência de visita entre os grupamentos em S_0 seja: V_1, ..., V_I, V_J, ...,V_M. O proposto é, dado um par de grupamentos adjacentes, alterar os vértices de

saída e entrada definidos nos grupamentos e refazer os caminhos nos dois grupamentos considerando os novos vértices escolhidos.

Seja:

- V_I e V_J: o par de grupamentos escolhido
- $v_i \in V_I$: o vértice de saída do grupamento V_I
- $v_j \in V_J$: o vértice de entrada do grupamento V_J

Além disso, uma aresta (v_s, v_t) é diferente de (v_i,v_j) quando $v_s \neq v_i$ e/ou $v_t \neq v_j$.

Considere as k_3 primeiras arestas diferentes de (v_i,v_j) que pertencem à lista de arestas $C_2(V_I, V_J)$, construída na fase anterior. Então, na solução S_0, (v_i,v_j) é substituída por cada aresta considerada em $C_2(V_I, V_J)$ e os caminhos hamiltonianos dos grupamentos V_I e V_J são refeitos considerando a substituição da aresta (v_i,v_j). Ou seja, suponha que $(v_i, v_j) \in S_0$ será substituída por $(v_s,v_t) \in C_2(V_I, V_J)$, os novos caminhos devem ser construídos considerando v_s como o novo vértice de saída do grupamento V_I e v_t, o novo vértice de entrada do grupamento V_J.

Deste modo, serão encontradas k_3 soluções que diferem de S_0 somente pela seqüência de visita nos nós dos grupamentos V_I e V_J. Sendo que, somente a melhor solução deve ser armazenada. O valor de k_3 é um dado fornecido ao algoritmo e deve ser definido de acordo com o problema.

É importante destacar que, a modificação das arestas é sempre realizada na solução S_0. Portanto, no exemplo anterior, se a troca da aresta (v_i,v_j) pela aresta (v_s,v_t) resultar em uma solução melhor que S_0, a solução encontrada deve ser apenas armazenada e a próxima aresta alterada deve pertencer a S_0.

Foram implementadas duas versões do algoritmo. As duas versões diferem-se pelo número de pares de grupamentos adjacentes em S_0 que são analisados na busca local.

Na primeira versão, apenas um par de grupamentos adjacentes é analisado. Enquanto que, na segunda, todos os pares de grupamentos adjacentes são analisados.

A seguir, é descrita a maneira como o par de grupamentos a ser analisado é escolhido na primeira versão do algoritmo:

Para cada grupamento V_I, resolver a instância do Problema do Caixeiro Viajante, PCV_I, que é definida pelos vértices pertencentes a V_I. No algoritmo, o PCV_I foi solucionado usando a Heurística da Inserção mais Próxima obtendo um ciclo hamiltoniano para cada grupamento definido no PCVG.

Seja:

- T_I = custo do ciclo hamiltoniano obtido no grupamento V_I (solução do PCV_I) e
- C_I = custo do caminho hamiltoniano obtido na fase de construção em cada no grupamento V_I.

Para cada par de grupamentos adjacente (V_I,V_J) definir S_{IJ} como:

$$S_{IJ} = (T_I - C_I) + (T_J - C_J)$$

O par de grupamentos considerado na primeira versão do algoritmo é selecionado escolhendo o par de grupamentos adjacentes (V_I,V_J) com menor soma.

O critério de escolha do par de grupamentos na fase de busca da segunda versão do GRASP padrão foi definido desta maneira, pois para cada par de grupamentos adjacentes, V_I e V_J, quanto menor a soma S_{IJ}, menor a diferença entre $T_I - C_I$ e $T_J - C_J$, sendo maior o custo do caminho hamiltoniano obtido em V_I e V_J. Deste modo, os grupamentos escolhidos na busca local da primeira versão do GRASP padrão para o PCVG são aqueles nos quais, o custo do caminho construído na fase de construção precisa ser melhorado.

A seguir, nas Figuras 6.2 e 6.3 é apresentado o pseudocódigo para o GRASP.

Algoritmo GRASP padrão para o PCVG

Início

(1) Defina a distancia máxima entre todos os pares de grupamentos $(V_I, V_J) \in V$;

(2) Enquanto critério de parada não atendido faca

Fase de Construção da solução S_0

(3) *Escolha um grupamento* V_1 *aleatoriamente*

(4) $Seq = (V_1)$; $V_{atual} = V_1$

(5) *Enquanto Seq não contém todos os grupamentos do PCVG faça*

(6) *Elabore* $C_1(V_{atual})$;

(7) *Se* C_1 *é vazia*

(8) *Então todos os grupamentos pertencem a Seq, a seqüência de grupamentos está definida*

(9) *Senão*

(8) *Elabore* $RCL_1(V_{atual})$, *onde* $| RCL_1(V_{atual}) | = k_1$ *com* $k_1 \in N$;

(8) *Escolha aleatoriamente um grupamento* V_J *da lista* $RCL_1(V_{atual})$

(9) *Insira* V_J *em Seq depois de* V_{atual};

(10) $V_{atual} = V_J$;

(12) *Fim-se;*

(13) *Fim-Enquanto;*

(14) *Para cada par de grupamentos vizinhos* V_1 *e* V_J *em Seq faca*

(15) *Elabore* $C_2(V_I, V_J)$;

(16) *Elabore* $RCL_2 (V_1, V_J)$, *onde* $| RCL_2 (V_1, V_J) | = k_2$;

(17) *Escolha aleatoriamente uma aresta* (v_i, v_j) *da* RCL_2 *para ser a aresta entre os grupamentos* V_1 *e* V_J;

(18) *Fim-Para;*

(19) *Para cada grupamento* V_1 *do PCVG faça*

(20) *Construa o caminho mínimo* P_1 *que visita todos os vértices de* V_1 *considerando como vértices final e inicial do caminho, aqueles que foram escolhidos como entrada e saída do grupamento* V_I;

(22) *Fim-Para;*

(23) $S^* = S_0$

Figura 6.2- GRASP padrão para o PCVG: Fase de Construção

O algoritmo continua na Figura 6.3, onde está descrito o pseudocódigo da fase de busca local.

Fase de Busca Local

(24) *Enquanto numero de pares de grupamentos analisados* $< \theta$

(25) *Para cada par de grupamentos analisado em* S_0, (V_I, V_J), *faça:*

Seja v_i *e* v_j *os vértices de saída e entrada dos grupamentos* V_I *e* V_J , *respectivamente*

(26) $i = 0$;

(27) *Enquanto* $i < K_3$ *faça*

(28) *Considere o i-ésimo elemento de* $C_2(V_I, V_J)$: *a aresta* (v_s, v_t) *com* $v_s \in V_I$ *e* $v_t \in V_J$

Em S_0, *troque a aresta* (v_i, v_j) *pela aresta* (v_s, v_t)

(29) *Construa os caminhos* P'_I *e* P'_J *nos grupamentos* V_I *e* V_J *considerando a nova aresta, obtendo uma nova solução* S'

(30) *Se* S' *é melhor que* S^* *então* $S^* = S'$;

(31) *Volte para solução original* S_0;

(32) *Fim-Enquanto;*

(33) *Fim-Para;*

(34) *Fim-Enquanto;*

(35) *Fim-Enquanto;*

Fim

Figura 6.3- Continuação do GRASP padrão para o PCVG: Fase de Busca Local

O loop, descrito pela linha (2) da Figura 6.2 a linha (35) da Figura 6.3, descreve cada iteração do GRASP padrão. A fase de construção é representada pelas linhas (2)-(22) da Figura 6.2. O problema PEG é resolvido nas linhas (4)-(13) da Figura 6.2, já nas linhas (14)-(22), é descrita a solução do PIG. Ainda na Figura 6.2, linha (23), a variável que representa a melhor solução encontrada, S*, é inicializada com a solução obtida na fase de construção. A fase de busca local esta representada na Figura 6.3, linhas (24)-(34). Na linha (24) da Figura 6.3, o valor de θ varia de acordo com a versão do GRASP padrão. Na primeira versão, foi usado $\theta = 1$ e na segunda, $\theta = M$, sendo que M é o número de grupamentos do grafo. Na i-ésima iteração do loop descrito pelas linhas (27)-

(32) da Figura 6.3, a aresta (v_i,v_j) de S_0 é trocada pela i-ésima aresta pertencente à C_2, o loop é executado até que sejam construídas k_3 soluções diferentes de S_0. Finalmente, na linha (30) da Figura 6.3, é armazenada a melhor solução obtida pelo algoritmo.

6.2. GRASP híbrido para o PCVG

Observando os resultados preliminares obtidos com a aplicação das heurísticas de construção, descritas no capítulo anterior, associadas às idéias de CHISMAN (1975), constatou-se que as soluções obtidas com as variações geradas para uma determinada instância do PCVG, em alguns casos, foram melhores se comparadas à solução obtida quando da aplicação dessas heurísticas na instância original. Portanto, neste algoritmo, a idéia foi resolver o PCVG segundo o método proposto por CHISMAN (1975) gerando variações (perturbações dos dados) da instância original e aplicando um algoritmo híbrido baseado nos princípios básicos apresentados no GRASP e na heurística GENIUS.

Como no GRASP padrão, descrito anteriormente, o segundo algoritmo proposto constitui-se de duas fases, uma fase de construção e uma fase de busca local. No GRASP padrão, a cada nova iteração, uma solução inicial é obtida na fase de construção e a fase de busca local começa imediatamente após esta solução inicial ser construída. Deste modo, o número de soluções construídas e o número de buscas realizadas são iguais ao número de iterações do GRASP. Entretanto, o GRASP híbrido consiste de apenas uma iteração onde, na fase de construção, são elaboradas I_C soluções e a fase de busca local começa somente quando todas as I_C soluções são construídas, considerando somente as I_B melhores soluções encontradas, com $I_B < I_C$.

Para facilitar o entendimento do que foi dito anteriormente, nas Figuras 6.4 e 6.5, estão descritos o pseudocódigo do GRASP padrão e do GRASP híbrido, respectivamente.

Comparação entre o GRASP padrão e o GRASP híbrido:

Algoritmo_Grasp_Padrão

(1) Faca i = 0 e Inicialize S (melhor solução);*
(2) Enquanto i < número de Iterações GRASP faça
(3) Fase de Construção(S_i);
(4) Fase de Busca Local(S');
(5) Se S' é melhor que S então atualiza(S*);*
(6) Fim-Enquanto;

Figura 6.4 - Comparação entre o GRASP padrão e o GRASP híbrido - Pseudocódigo para o GRASP padrão

Algoritmo_GRASP_Híbrido

(1) i = 0;
(2) Enquanto i < I_C (número máximo de soluções iniciais) faça
(3) Construa (S_i) (i-ésima solução);
(4) Fim-Enquanto;
(5) Inicialize(S);*
(6) j = 0 ;
(7) Enquanto j < I_B (número máximo de iterações da fase busca local) faça
(8) Realize Busca Local na Vizinhança da j-ésima melhor solução construída obtendo S';
(9) Se S' é melhor que S então atualize(S*);*
(10) Fim_Enquanto;

Figura 6.5 - Comparação entre o GRASP padrão e o GRASP híbrido - Pseudocódigo para o GRASP Híbrido

Nas duas figuras, S* representa a melhor solução encontrada. No algoritmo Grasp Padrão, descrito na Figura 6.4, cada iteração está representada pelas linhas (2)-(6). Na i-ésima iteração, a solução S_i é construída na linha (3) da Figura 6.4 e a busca local, realizada na vizinhança de S_i, é representada pela linha (4) da mesma figura,

finalizando a iteração. Na Figura 6.5 que representa o GRASP híbrido, a cada iteração da fase de construção, representada pelas linhas (2)-(4) da figura, uma solução é construída. São realizadas I_C iterações na fase de construção. Na fase de busca local, descrita nas linhas (7)-(10) da Figura 6.5, a cada iteração é considerada uma solução construída na fase anterior. A fase de busca começa somente quando as I_C soluções inicias são construídas e termina quanto as I_B melhores soluções são analisadas.

6.2.1. Fase de Construção do GRASP híbrido

As soluções obtidas na fase de construção são elaboradas aplicando o procedimento de inserção, GENI, na instância original do problema e nas α variações geradas (perturbações do tipo P2).

Seja:

- $PCVG_G$: a instância original do PCVG
- PCV_G: a instância do PCV definida a partir da instância do PCVG
- $PCVG_I$: i-ésima variação do $PCVG_G$
- d_{ij}: custo da aresta (v_i,v_j) no $PCVG_G$
- d^I_{ij}: custo da aresta (v_i,v_j) na i-ésima variação $PCVG_I$

Na primeira iteração da fase de construção, a primeira solução é elaborada considerando o $PCVG_G$, ou seja, para as arestas que conectam vértices do mesmo grupamento, os custos considerados são os da instância original. Nas demais iterações, as próximas soluções são construídas considerando as variações que foram geradas. Neste caso, para as arestas que conectam vértices do mesmo grupamento, os custos considerados são os da variação. Deste modo:

I_C (número de iterações da fase de construção) = α (número de variações) + 1.

Com as I_C soluções construídas é definida uma lista, denominada $C(PCVG_G)$, ordenando de forma crescente os custos das soluções obtidas.

Seja S_I, a solução obtida a partir da variação $PCVG_I$. Cada solução S_I determina dois custos, C_I e C'_I:

- C_I:custo de S_I considerando os custos das arestas no $PCVG_G$

- C'_I:custo de S_I considerando os custos das arestas nas variações

Na elaboração da $C(PCVG_G)$, o custo considerado para ordenar a lista quando da inclusão de uma solução S_I é C_I. Portanto, a cada iteração da fase de construção, o custo da solução obtida pela aplicação do GENI nas variações é refeito considerando os custos originais das arestas para que S_I se torne uma solução para o problema original. S_I é modificada substituindo, para cada aresta (v_i,v_j) que pertence a S_I, o custo d^I_{ij} por d_{ij}.

Com os I_B primeiros elementos da lista $C(PCVG_G)$ é elaborada a lista de soluções candidatas à solução do $PCVG_G$, denominada $RCL(PCVG_G)$. Desta forma, $RCL(PCVG_G)$ contém apenas as I_B melhores soluções obtidas.

Na Figura 6.6, está representado o pseudocódigo para a fase de construção do algoritmo, com p representando o tamanho da vizinhança considerada no GENI.

Na linha (1) da Figura 6.6, i é inicializada. A variável i é o contador de soluções. O loop, descrito pelas linhas (2)-(12) da Figura 6.6, representa as iterações da fase de construção. Ainda na mesma figura, linha (3), caso seja a primeira iteração, o PCV_G é gerado a partir da instância original do PCVG. Caso contrário, o PCV_G é gerado a partir de uma das variações do PCVG. Na linha (7), a solução é construída aplicando GENI no PCV_G. Nas linhas (8)-(9) da Figura 6.6, o custo C_I é obtido para ordenar as soluções na

C(PCVG$_G$). Na linha (9), S_I é inserida na lista C(PCVG$_G$). Enquanto que, na linha (10), a solução S_I é armazenada com o custo C'$_I$. Finalmente, na linha (12), a fase de construção termina com a elaboração da RCL(PCVG$_G$).

```
    Algoritmo_FaseConstrução_GRASP_Híbrido
Inicio
  (1)   i = 0;
  (2)   Enquanto i < I_C faça
  (3)       se i = 0
  (4)           Elabore PCV_G a partir do PCVG_G ;
            senão
  (5)           Elabore PCV_G a partir do PCVG_i ;
  (6)       fim-se;
  (7)       Construa(S_I) aplicando GENI;
  (8)       se i > 0
                para cada aresta (v_i,v_j) de S_I, tal que, v_i e v_j pertencem ao mesmo
                grupamento faça
                                C_I = C'_I - d^I_ij + d_ij
                fim-para;
  (9)       fim-se
  (10)      Insira S_I em C(PCVG_G) segundo o custo C_I;
  (11)      Armazene(S_I,C'_I);
  (12)  Fim-Enquanto;
  (13)  Elabore LCR(PCVG_G);
Fim
```

Figura 6.6 - Pseudocódigo - Fase de Construção do GRASP híbrido usando GENI

6.2.2. Fase de Busca Local do GRASP híbrido

Na j-ésima iteração da fase de busca local do GRASP híbrido, é realizada uma busca local na vizinhança da j-ésima solução pertencente à RCL(PCVG$_G$). O algoritmo de busca local usado é o US-M que foi definido no capítulo anterior. A fase de busca local termina quanto todas as soluções da RCL(PCVG$_G$) são analisadas.

Seja S_J, a j-ésima solução da RCL($PCVG_G$) e suponha que S_J seja uma solução obtida a partir da variação $PCVG_J$. Na j-ésima iteração da fase de busca local, o procedimento US-M deverá ser aplicado em S_J considerando os custos das arestas nas variações e somente depois de obtida a solução, o custo deve ser modificado de forma que os custos originais das arestas sejam considerados e a solução seja válida para a instância original. Por isso, na fase de construção, a cada iteração, a solução S_I é armazenada associada ao custo C'_I, que é o custo de S_I considerando os custos arestas nas variações.

O pseudocódigo da fase de busca local é apresentado na Figura 6.7.

```
Algoritmo_BuscaLocal_GRASP_Híbrido
Início
(1)  Inicialize S*;
(2)  j = 0;
(3)  Enquanto j < I_B faça
(4)        Encontre S' aplicando US-M na j-ésima solução da LCR(PCVG_G): S_J;
(5)  Se S_J foi obtida a partir de uma variação
                para cada aresta (v_i,v_j) de S_J onde v_i e v_j pertencem ao mesmo
                grupamento faça
                        C_I = C'_I - d^J_ij + d_ij
                fim-para;
(6)  Fim-se;
(7)  Se S' é melhor que S*
                então S*=S';
(8)  Fim-se;
(9)  Fim-Enquanto;
(10) Modifique S* para que se torne uma solução do PCVG_G
Fim;
```

Figura 6.7 - Pseudocódigo - Fase de Busca Local do GRASP híbrido usando US-M

Na linha (1) da Figura 6.7, S*, que representa a melhor solução encontrada, é inicializada. As iterações da fase de busca local estão representadas na Figura 6.7 pelas linhas (3)-(9). Na linha (4) da mesma figura, o procedimento US-M é aplicado nas

soluções que pertencem a RCL($PCVG_G$). As soluções obtidas a partir de variações do $PCVG_G$ são modificadas nas linhas (5)-(6) da Figura 6.7. Em (7), se a solução obtida for melhor que S*, então S* é atualizada. No final do algoritmo, a melhor solução obtida é solução da instância do PCV. Portanto, na Figura 6.7, linha (10), S* é modificada de maneira que se torne uma solução para a instância original. Essa modificação é realizada diminuindo L dos custos das arestas pertencentes a S* que conectam vértices de grupamentos diferentes.

O que justifica o fato da busca local no GRASP híbrido não ser realizada imediatamente após a construção de cada solução como no GRASP padrão é a escolha do procedimento GENI, como o método de construção de soluções, e do procedimento US-M, como método de busca. Pois o tempo de execução do procedimento GENI para construir cada solução é bem menor que o tempo gasto pelo algoritmo US-M para executar a busca local na vizinhança da solução. Portanto, considerando um número maior de variações na fase de construção, pode-se obter uma variedade de soluções iniciais e realizar a busca somente nas melhores.

O fator aleatório do algoritmo está na maneira como as variações da instância do PCVG são geradas. Como descrito no capítulo anterior, os custos das arestas nas variações são escolhidos aleatoriamente entre um conjunto de valores. Além disso, na fase de construção da heurística GENIUS, GENI, o próximo vértice a ser inserido na solução é escolhido aleatoriamente.

Como foi descrito no pseudocódigo do algoritmo, a cada solução construída a partir de uma variação, o custo da solução é modificado para elaborar a lista C($PCVG_G$). Além disso, na fase de busca local, a cada iteração, os custos das soluções obtidas com a aplicação do US-M nas variações também são modificados para atualizar a variável S*.

Esta modificação do custo, a cada iteração da fase de construção e da fase de busca local, pode aumentar o tempo de execução do algoritmo.

Com o objetivo de otimizar o tempo de execução, foi proposta outra versão, onde a idéia foi modificar o custo somente na fase de busca local quando da atualização da variável S*, ou seja, a elaboração da lista $C(PCVG_G)$ foi realizada de acordo com o custo da solução na variação. Nos testes preliminares realizados com as duas versões do algoritmo, constatou-se que as soluções obtidas com a segunda versão foram tão boas quanto as soluções obtidas com a primeira, porém exigindo um tempo de execução menor. Portanto, no próximo capítulo, serão apresentados somente os testes numéricos com a segunda versão.

6.3. VNS padrão para o PCVG

O próximo algoritmo proposto foi elaborado a partir das idéias da metaheurística VNS (MANDENOVIC *et. al.*,1997). O algoritmo possui duas fases, a fase de construção e a fase de busca local. Na fase de construção, o procedimento GENI é usado para construir uma solução inicial. Enquanto que, US é o procedimento usado na fase de busca local. Porém, como no VNS, o número de estruturas de vizinhanças consideradas na fase de busca local é maior que um, sendo elaboradas duas versões do VNS para o PCVG que diferem pela maneira como o procedimento US é aplicado na fase de busca. Na primeira versão, o procedimento de busca é denominado VNSC e na segunda versão, o procedimento de busca é denominado VNSR.

Seja:

- S_0: solução encontrada no método de construção e C: custo da solução S_0
- S*: melhor solução encontrada e C*: custo da solução S*
- p: tamanho da vizinhança considerada nos procedimentos GENI e US

- $PCVG_G$: instância original do PCVG.
- PCV_G: instância do problema do caixeiro viajante obtida a partir da $PCVG_G$
- k_{max}: tamanho máximo de p, k_{min}: tamanho mínimo de p
- k_{lim}:valor definido entre k_{min} e k_{max}, ou seja, $k_{min} < k_{lim} < k_{max}$

A Figura 6.8 representa o pseudocódigo da primeira versão do VNS.

```
Primeira_Versão_VNS_PCVG
Inicio
(1) Obtenha o PCV_G;
(2) Construa(S_0) aplicando GENI com p = k_max;
(3) S = S* = S_0;
(4) Faça k = k_mim;
(5) Enquanto k ≤ k_lim
(6)     Aplique US em S com p = k obtendo S'
(7)     se S'< for melhor S*
        então
                S* = S';
                S = S';
                k = k_min;
        senão k = k +1;
(8) Fim-Enquanto;
(9) Atualize (S*);
Fim;
```

Figura 6.8- Pseudocódigo para a primeira versão do VNS padrão para o PCVG

Na linha (1) da Figura 6.8, é obtida a instância do PCV (PCV_G). Na linha (2) da mesma figura, está representada a fase de construção onde a solução inicial é obtida aplicando o procedimento GENI com p igual a k_{max}. S* é inicializada na linha (3) da Figura 6.8. Nas linhas (5)-(8), estão descritas as iterações do VNSC. A busca começa aplicando o procedimento US na solução encontrada pelo GENI com p = k_{min}, que representa a primeira vizinhança. Seja S, a solução que esta sendo considerada na busca e S', a solução encontrada com a aplicação de US em S, representada na linha (6) da Figura 6.8. Na linha (7), se S' for melhor que S*, então S* é atualizada e a busca

continua aplicando US em S' com p = k_{min} (primeira vizinhança). Por outro lado, caso S' não seja melhor que S*, o procedimento US é aplicado novamente em S, porém aumentando o tamanho da vizinhança considerada (nova vizinhança). A fase de busca local termina quando p > k_{lim}. A solução S*, obtida no VNSC, é solução do PCV_G. Portanto, na linha (9) da Figura 6.8, a solução S* é modificada de maneira que se torne uma solução para a instância original do PCVG.

A seguir, a segunda versão do VNS está descrita na Figura 6.9.

```
Segunda_Versão_VNS_PCVG
Inicio
(1) Obtenha o PCV_G;
(2) Construa(S_0) aplicando GENI com p = k_max;
(3) S = S* = S_0;
(4) Faça k = k_min;
(5) Enquanto k ≤ k_lim
(6)     Faça t =0;
(7)     Enquanto t < N
(8)         Tente retirar o vértice v_t e reinseri-lo no ciclo usando os dois tipos de
            retirada e inserção proposto em US e GENI. Seja S' a solução encontrada
(9)         Se S' for melhor que S* então
                Faça S*:= S'; k = k_min
(10)        Aplique US em S' considerando todos os vértices da solução;
                Seja S", a solução encontrada
(11)        Se S"for melhor que S* então S* = S";
(12)        Faça k = k + 1; Volte ao passo (5) com S";
(13)    senão   Faça t = t+1;
(14)        Se t = n+1 então
(15)            Faça k = k + 1;
(16)            vá para o passo (5) ;
(17)        senão volte ao passo (7);
(18) Fim-Enquanto
(19) Fim-Enquanto;
(20) Atualize(S*);
Fim;
```

Figura 6.9- Pseudocódigo para a segunda versão do VNS padrão para o PCVG

Assim como na primeira versão do VNSP, na fase de construção da segunda versão, representada pela linha (2) da Figura 6.9, a solução é construída pelo procedimento GENI com $p = k_{max}$. As linhas (5)-(20) da Figura 6.9 representam as iterações do VNSR. Seja S, a solução que esta sendo considerada na busca. Na linha (8) da Figura 6.9, o vértice v_t é retirado e reinserido de S obtendo um novo ciclo S'. Se S' for melhor que S*, então na linha (9), S* é atualizada e, na linha (10), o procedimento US é aplicado em S' com $p = k_{min}$ obtendo S". Ainda na Figura 6.9, linha (11), caso S" seja melhor que S*, S* é atualizada. Na linha (12), a busca continua com S" aumentando o valor de p. Por outro lado, na linha (9), caso S' não seja melhor que S*, então o próximo vértice de S, v_{t+1}, é retirado e reinserido no ciclo, ou seja, a busca local continua na solução S. Neste ponto, se todos os vértices de S já foram reinseridos e não foi possível obter nenhuma melhora, então a busca continua em S, mas aumentando o valor de p. A fase de busca termina quando $p > k_{lim}$. No próximo capítulo, são apresentados os resultados computacionais obtidos com as duas versões do algoritmo.

6.4. VNS híbrido para o PCVG

O último algoritmo proposto é um outro algoritmo metaheurístico híbrido que combina o VNS à heurística GENIUS. Entretanto, algumas idéias propostas na metaheurística GRASP são usadas para tentar encontrar novas soluções. O algoritmo possui duas fases, uma fase de construção e uma fase de busca local. Assim como no GRASP híbrido, para resolver uma instância do PCVG, são consideradas variações geradas a partir da instância original perturbando os dados de entrada (perturbação do tipo P2). A principal diferença entre eles está no algoritmo usado na fase de busca local. No GRASP híbrido, o procedimento usado na busca local é o US-M. Já no VNS híbrido, o procedimento de busca local é o mesmo usado na segunda versão do VNS padrão, VNSR.

Seja $PCVG_G$, a instância original do PCVG e $PCVG_I$, a i-ésima variação do $PCVG_G$. Na fase de construção, I_C soluções para o $PCVG_G$ são construídas com GENI, sendo que, α soluções são construídas baseadas nas α variações geradas para o $PCVG_G$. Assim como no GRASP híbrido, com as soluções encontradas é definida a lista $C(PCVG_G)$ e com as I_B melhores soluções de $C(PCVG_G)$ é elaborada a $RCL(PCVG_G)$. A única diferença entre a fase de construção dos dois algoritmos está no tamanho da vizinhança considerada no GENI. No GRASP híbrido, foi usado $p = k_{lim}$ e no VNS híbrido, $p = k_{max}$.

Na fase de busca local, é realizada uma busca local na vizinhança das I_B soluções pertencentes à lista $RCL(PCVG_G)$. A Figura 6.10 apresenta o pseudocódigo para o VNS híbrido proposto.

Algoritmo_VNS_Híbrido

Início

(1) i = 0;

(2) Enquanto $i < I_C$ faça

(3) Se i = 0 então

obtenha o PCV_G através do $PCVG_G$

senão

obtenha o PCV_G através do $PCVG_I$

(4) Construa(SI) aplicando GENI com p = kmax;

(5) Insira S_I na lista $C(PCVG_G)$;

(6) Fim-Enquanto;

(7) Elabore $RCL(PCVG_G)$

(8) J = 0;

(9) Enquanto $J<I_B$ faça

(10) $k = k_{min}$;

(11) Aplique VNSR em S_J (j-ésima solução da $RCL(PCVG_G)$) e encontre S'

(12) Se S_J foi obtida a partir de uma variação

para cada aresta (v_i,v_j) de S_J, tal que, v_i e v_j pertencem ao mesmo grupamento faça

$$C_J = C'_J - d^J_{ij} + d_{ij}$$

fim-para;

(13) Fim-se;

*(14) Se S' é melhor que S**

então S := S';*

(15) Fim-se;

(16) Fim-Enquanto;

(17)Modifique(S);*

Fim;

Figura 6.10- Pseudocódigo – VNS híbrido para o PCVG

Nas linhas (3)-(6) da Figura 6.10, estão descritas as I_C iterações da fase de construção. Na linha(4), GENI é aplicado com $p = k_{max}$ e na linha (5), a lista $C(PCVG_G)$ é elaborada ordenando as soluções de forma crescente segundo os seus custos. Como no GRASP híbrido, caso a solução tenha sido obtida através de uma variação, o custo

usado para elaborar a lista $C(PCVG_G)$ é o custo da solução na variação. Na linha (7) da Figura 6.10, as I_B melhores soluções de $C(PCVG_G)$ formam a $RCL(PCVG_G)$. As linhas (9)-(16) da Figura 6.10 descrevem a fase de busca local, onde na iteração j, linha (11), o algoritmo VNSR é aplicado na j-ésima solução da lista $RCL(PCVG_G)$ obtendo S'. Se S_J foi obtida através de uma variação, seu custo é modificado na linha 12 da Figura 6.10. Ainda na Figura 6.10, linha (14), caso S' seja melhor que S*, a solução S* é atualizada. S* e solução da instância do PCV definida a partir da instância do PCVG, portanto, na linha (17), S* é modificada para que se torne uma solução da instância original do PCVG.

7. Resultados

Neste capítulo, são apresentados os resultados computacionais obtidos com os algoritmos propostos nos capítulos 6 e 7. Todos os algoritmos foram implementados usando a linguagem C.

Os algoritmos e suas respectivas versões são descritos resumidamente na Tabela 1. O nome dado aos algoritmos é apresentado na primeira coluna da Tabela 1. Enquanto que, na segunda, terceira e quarta coluna, é apresentada uma breve descrição do algoritmo e os métodos usados nas fases de construção e de busca local de cada algoritmo, respectivamente. Cada linha da Tabela 1 corresponde a um algoritmo. Vale ressaltar que, todos os algoritmos da Tabela 1, com exceção de H1, foram propostos neste trabalho.

Nome do Algoritmo	Descrição	Fase de Construção	Fase de Busca Local
HIMP	Obtém o PCV através do PCVG. O PCV é solucionado por umas das heurísticas de construção implementadas. A solução obtida é transformada em uma solução do PCVG	Heurística da Inserção Mais Próxima	Não possui fase de Busca Local
HIMB	Obtém o PCV através do PCVG. O PCV é solucionado por umas das heurísticas de construção implementadas. A solução obtida é transformada em uma solução do PCVG	Heurística da Inserção Mais Barata	Não possui fase de Busca Local
HVMP	Obtém o PCV através do PCVG. O PCV é solucionado por umas das heurísticas de construção implementadas. A solução obtida é transformada em uma solução do PCVG	Heurística do Vizinho Mais Próximo	Não possui fase de Busca Local
H1 (GENDREAU *et al.* 1996b)	Obtém o PCV através do PCVG. O PCV é solucionado pela heurística GENIUS. A solução obtida é transformada em uma solução do PCVG	GENI	US
H1M	Transforma o PCVG no PCV. O PCV é solucionado pela heurística GENIUS-M. A solução obtida é transformada em uma solução do PCVG	GENI	US-M
GRASP1	Primeira versão do algoritmo GRASP Padrão para o PCVG	Método de Construção definido para o Grasp padrão	Método de Busca Local definido para a primeira versão do GRASP padrão
GRASP2	Segunda versão do algoritmo GRASP Padrão para o PCVG	Método de Construção definido para o Grasp padrão	Método de Busca Local definido para a segunda versão do GRASP padrão
GRASP3	Algoritmo GRASP Híbrido para o PCVG	GENI + Variações	USM + Variações
VNS1	Segunda versão do VNS padrão para o PCVG	GENI	VNSR
VNS2	Primeira versão do VNS padrão para o PCVG	GENI	VNSC
GRASP4	VNS Híbrido para o PCVG	GENI + Variações	VNSR + Variações

Tabela 1- Algoritmos elaborados para o PCVG

As instâncias usadas nos testes foram geradas da seguinte forma:

Seja

- $G = (V_G, E_G)$: instância onde o PCVG é definido
- V_G: conjunto de vértices de G
- E_G: conjunto de arestas de G
- $N = |V_G|$ = número de vértices de G
- $(v_i, v_j) \in E_G$ com $v_i, v_j \in V_G$
- $d_{ij} \in N^+$, o custo da aresta (v_i, v_j)
- $V_G = V_1 \cup V_2 \cup, \ldots, \cup V_M$: os grupamentos do grafo G
- M = número de grupamentos de G
- $C_{max} = \max\{d_{ij}\}$
- m_I: número mínimo de vértices pertencentes ao grupamento V_I

Dado N e C_{max}, a cada aresta (v_i, v_j), d_{ij} foi determinado escolhendo aleatoriamente um número inteiro no intervalo $[1, C_{max}]$. Além disso, os vértices foram associados aleatoriamente aos grupamentos na ordem em que foram gerados, de maneira que, em cada grupamento V_I, $m_I > 0$.

7.1. Resultados obtidos com o GRASP padrão

Nesta seção, são apresentados os resultados obtidos com as duas versões do GRASP padrão (GRASP1 e GRASP2).

Na Tabela 2, os resultados obtidos com GRASP1 são comparados aos resultados obtidos com GRASP2. Antes de executar os testes apresentados, foi preciso definir os melhores valores para os parâmetros k_1, k_2 e k_3. Dentre os valores testados para k_1, k_2 e k_3, os melhores resultados foram obtidos com $k_1 = 2$, $k_2 = 2$ e $k_3 = 4$. Essa análise foi

k_3, os melhores resultados foram obtidos com $k_1 = 2$, $k_2 = 2$ e $k_3 = 4$. Essa análise foi feita considerando o tempo de execução e custo das soluções. Portanto, na Tabela 2, os resultados foram obtidos com k_1, k_2 e k_3 iguais a 2, 2 e 4, respectivamente. Nos dois algoritmos, foram realizadas 1000 iterações. Foram testadas instâncias com 50, 100, 200, 300 e 500 nós e com número de grupamentos iguais a 10% e 20% do número de vértices.

Na primeira coluna da Tabela 1, são descritos o número de vértices e o número de grupamentos da instância do PCVG. Na segunda e terceira coluna, são apresentados o custo e o tempo médios obtidos pela primeira e segunda versão do GRASP padrão, GRASP 1 e GRASP2, respectivamente. Sendo que, para cada instância gerada, os algoritmos foram executados 10 vezes. Nas linhas das tabelas, são apresentados os resultados obtidos para cada instância. Os testes foram realizados em uma Sun Ultra1, 128 Mb de memória, sistema operacional SunOS 5.7.

		GRASP1		GRASP2	
N	**M**	**Custo**	**Tempo**	**Custo**	**Tempo**
50	**5**	218	14.13 s	214	39.43s
	10	310	3.75s	302	13.73s
100	**10**	961	37.40s	931	2m23.81s
	20	1408	10.64s	1376	49.25s
200	**20**	3820	2m7.54s	3751	11m12.72s
	40	5657	40.32s	5550	3m22.85s
300	**30**	8846	4m0.87s	8805	23m51.09s
	60	12692	1m27.62s	12539	8m17.41s
400	**40**	16378	7m14.15s	16167	39m31.51s
	80	22456	2m46.06s	22305	14m19.36s
500	**50**	25138	11m8.69s	24810	64m36.46s
	100	34936	4m24.64s	34627	24m8.71s

Tabela 2 – Resultado do GRASP1 e GRASP2

As soluções encontradas pelo GRASP2 foram melhores que as encontradas com GRASP1. Porém, no GRASP1, o tempo de execução foi menor. A diferença entre o GRASP1 e o GRASP2, em relação ao custo das soluções obtidas e ao tempo de execução, se justifica pelo tamanho da vizinhança considerada nos dois algoritmos. Na

fase de busca do GRASP1, o número de soluções vizinhas analisadas é menor que no GRASP2.

Tanto no GRASP1 quanto no GRASP2, dado um número de vértices, o tempo de execução foi menor nas instâncias onde o número de grupamentos foi maior. Nessas instâncias, o número de vértices em cada grupamento é menor, sendo assim, o problema de encontrar um caminho mínimo definido entre os vértices de cada grupamento também é menor.

Os resultados obtidos com o GRASP padrão foram comparados com H1M, que foi executado com p (tamanho da vizinhança considerada no GENI e US) = 4. Na Tabela 3, são apresentados os resultados do GRASP2 e H1M.

Assim como na tabela anterior, na Tabela 3, a primeira coluna contém o número de vértices e o número de grupamentos das instâncias. Porém, na segunda e terceira coluna, são apresentados o custo e o tempo médios obtidos com H1M e GRASP2, respectivamente.

Além disso, os resultados apresentados na Tabela 3 foram obtidos executando 10 vezes o GRASP2 e o H1M para cada instância em uma máquina Sun Ultra1, 128 Mb de memória, sistema operacional SunOS 5.7.

		H1M		**GRASP2**	
N	**M**	**Custo**	**Tempo**	**Custo**	**Tempo**
100	**10**	877	9.51s	931	2m23.81s
	20	1275	9.21s	1376	49.25s
200	**20**	3300	1m0.55s	3751	11m12.72s
	40	5047	1m6.61s	5550	3m22.85s
300	**30**	7547	2m49.48s	8805	23m51.09s
	60	10611	5m37.04s	12539	8m17.41s
500	**50**	20342	26m11.45s	24810	64m36.46s
	100	27853	22m12.99s	34627	24m8.71s

Tabela 3- GRASP2 x H1

Os resultados obtidos com o GRASP2 foram inferiores àqueles obtidos com H1M. Uma justificativa para esse resultado está na maneira como os dois problemas definidos no PCVG, PEG e PIG, são resolvidos. No GRASP2, o PEG é resolvido e com a solução

do PEG, o PIG é solucionado. Já em H1M, a instância do PCVG é transformada em uma instância do PCV que é solucionado pela heurística GENIUS-M. A solução obtida é modificada de maneira que se torne uma solução para o PCVG. Portanto, em H1M os dois problemas são resolvidos simultaneamente, sendo que a seqüência de visitas entre os grupamentos é calculada de acordo com a seqüência determinada entre os vértices do grafo, obtendo melhores soluções.

7.2. Resultados obtidos com as Heurísticas elaboradas para o PCVG

Nas Tabelas 7 a 17, são apresentados os resultados obtidos com os algoritmos baseados nas heurísticas de construção e com a heurística H1M comparados aos resultados obtidos com H1. Nos algoritmos, a instância do PCVG foi resolvida como uma instância do PCV e depois, a solução obtida foi modificada em uma solução para o PCVG. Além disso, foram geradas α variações da instância original do PCVG obtendo α soluções para o problema. Nos testes, o valor de $\alpha = 50$ e nos algoritmos H1 e H1M, o tamanho da vizinhança considerada (p) foi igual a 4.

Os algoritmos foram testados para instâncias de 50 até 1000 nós. Os nomes dados às instâncias geradas foram escolhidos de acordo com o número de vértices. A instância com 100 vértices foi denominada G100, com 200 vértices, G200, e assim por diante até a instância com 1000 vértices, denominada G1000. As 50 variações geradas para cada instância foram denominadas V1, ..., Vα. Além disso, para uma determinada instância do PCVG, o número de grupamentos foi igual a 10% do número de vértices da mesma instância, ou seja, para a instância G100, M = 10. Para G200, M = 20, até G1000, onde M= 100.

Cada tabela contém os resultados obtidos para uma determinada instância. Na Tabela 7, são apresentados os resultados obtidos com a instância G50 e suas 50

variações. A Tabela 8 contém os resultados obtidos com a instância G100 e suas variações e assim até a Tabela 17, que apresenta os resultados obtidos com a instância G1000 e suas variações.

Nas Tabelas 7 a 17, os nomes dados às instâncias do PCVG são apresentados na primeira coluna. Nas colunas 2, 3, 4, 5 e 6 são representados o custo e o tempo obtido com as heurísticas H1M, H1, HIMP, HIMB e HVMP, respectivamente. O tempo foi determinado calculando a média dos tempos de execução dos algoritmos na instância original e nas variações. A primeira linha das tabelas descreve os resultados obtidos com a instância original do PCVG e as demais linhas descrevem os resultados obtidos com as variações.

Os resultados, descritos nas Tabelas 7 a 17, foram realizados em um PC com processador Pentium III, 600 MHz, 520 MB de memória e sistema operacional Linux.

Esses resultados são resumidos na Tabela 4. As linhas da Tabela 4 correspondem às instâncias do PCVG. Para cada instância, na primeira coluna, é apresentado o algoritmo que obteve melhor solução. Na segunda coluna, são apresentados os algoritmos onde a melhor solução foi encontrada através de uma das variações da instância original. Na última coluna, os algoritmos HIMP, HIMB e HVMP são apresentados ordenados segundo o custo da solução obtida.

Por exemplo, considere a linha da Tabela referente à instância G50:

- Primeira coluna: H1M obteve melhor solução
- Segunda coluna: em todos os algoritmos a melhor solução foi obtida com uma das variações
- Terceira coluna: entre os algoritmos elaborados com as heurísticas de construção, HIMP foi melhor que HIMB que foi melhor HVMP

Grafo	Melhor Solução	Algoritmos onde as melhores soluções foram encontradas com as variações	HIMP x HIMB x HVMP
G50	H1M	Todos	HIMP/HIMB/ HVMP
G100	H1	Todos	HVMP/HIMB/HIMP
G200	H1M	Todos	HVMP/HIMB/HIMP
G300	H1	Todas	HVMP/HIMB/HIMP
G400	H1M	Todas	HIMP/HVMP/HIMB
G500	H1M	HIMP, HIMB e HVMP	HVMP/HIMB/HIMP
G600	H1M	HIMB	HVMP/HIMB/HIMP
G700	H1/H1M	HIMP e HVMP	HVMP/HIMB/HIMP
G800	H1M	HIMP e HIMB	HVMP/HIMB/HIMP
G900	H1M	HIMP, HIMB e HVMP	HVMP/HIMB/HIMP
G1000	H1M	HIMP, HIMB e HVMP	HVMP/HIMB/HIMP

Tabela 4 – Comparação entre os custos das soluções obtidas pelos algoritmos H1, H1M, HIMP, HIMB e HVMP

Portanto, nos testes realizados, com exceção das instâncias G100, G300 e G700, H1M foi melhor que H1, tanto na qualidade da solução obtida quanto em relação ao tempo de execução (o tempo de execução está apresentado nas tabelas 7 a 17). Em todas as instâncias, as soluções geradas a partir das variações foram melhores em pelo menos um dos algoritmos propostos. Em relação aos algoritmos implementados com as heurísticas de construção, HIMP, HIMB e HVMP, com exceção das instâncias G100 e G400, HVMP obteve melhores soluções exigindo um tempo computacional bem inferior ao dos outros algoritmos.

A diferença entre o tempo computacional exigido pelo H1 quando comparado ao H1M (tabela 7 a 17) se justifica pelos diferentes métodos adotados na busca, mesmo que em H1M, o método de busca definido, USM, tenha sido elaborado a partir do US que foi usado no H1. Isto porque o US começa escolhendo um vértice inicial para ser retirado e reinserido da solução. Feito isto, caso a solução obtida seja melhor, a busca recomeça na solução obtida, voltando ao vértice inicial escolhido. No USM, caso a solução encontrada na busca seja melhor que a atual, o algoritmo recomeça

considerando a solução encontrada, porém, a busca continua a partir do último vértice e não retorna ao vértice escolhido inicialmente.

7.3. Resultados obtidos com os algoritmos baseados no GRASP e VNS

Na Tabela 5, são apresentados os resultados obtidos com os algoritmos GRASP2, GRASP3, GRASP4, VNS1 e VNS2, comparados ao H1. Nas linhas da tabela, estão apresentados os resultados obtidos para cada instância gerada para o PCVG. Na primeira coluna, está descrito o nome da instância. Nas colunas 2, 3, 4, 5, 6 e 7, são apresentados os custos e os tempos obtidos com VNS1, VNS2, H1, GRASP3, GRASP4 e GRASP2. Para os algoritmos VNS1, VNS2, H1 e GRASP2, os resultados apresentados foram obtidos executando os algoritmos 10 vezes em cada instância gerada. Deste modo, são apresentados o custo e o tempo médios. Porém, para os algoritmos GRASP3 e GRASP4, o número de execução em cada instância foi igual a 1.

Os testes foram realizados em um PC com processador Pentium III, 650 MHz, 250 Mb de memória, sistema operacional Linux. O número de vértices das instâncias geradas variou de 50 a 1000 nós. Os nomes dados às instâncias foram definidos como descrito anteriormente.

No GRASP2 os parâmetros k_1, k_2 e k_3 foram iguais a 2, 2 e 4, respectivamente e o número de iterações foi igual a 1000. No GRASP3, I_C =100 e I_B = 50, com exceção da instância G1000, onde I_C=70 e I_B=30. No GRASP4, I_C = 50 e I_B = 20, com exceção da instância G900 onde I_B = 15, e G1000, onde I_B= 10. Nos algoritmos VNS1, VNS2 e GRASP4, $k_{min} = 2$, $k_{max} = 6$ e $k_{lim} = 4$. Além disso, no H1 e GRASP3, $p = 4$.

	VNS1		VNS2		H1		GRASP3		GRASP4		GRASP2	
	C	Tempo	C	Tempo	C	T	C	Tempo	C	Tempo	C	Tempo
G50	406	0.23s	406	0.21s	413	0.39s	**375**	13.97s	**372**	9.36s	469	6.12s
G100	749	2.04s	749	2.44s	760	2.38s	740	1m22.68s	**736**	1m0.77s	874	21.65s
G200	1634	11.56s	1614	22.15s	1631	22.98s	1624	10m01..84s	**1605**	7m20.65s	2003	1m23.84s
G300	2473	1m7.88s	2481	1m3.06s	2515	1m15.01s	2467	30m24.97s	**2444**	23m52.74s	3063	4m7.87s
G400	3013	2m26.55s	2996	3m3.3s	3027	4m45.19s	**2985**	74m55.70s	3000	69m02.83s	3857	8m49.69s
G500	4093	11m24.49s	4127	7m52.14s	4101	10m27.37s	4108	154m27.61s	**4089**	169m32.32s	5256	13m10.04s
G600	4730	13m33.66s	4730	13m4.83s	4789	23m41.93s	4735	323m37.01s	**4700**	362m25.64s	6153	20m11.96
G700	5814	46m36.48s	5830	36m30.89s	5894	51m27.30s	5858	493m13.99s	**5785**	658m45.82s	7468	25m43.7s
G800	6014	59m42.56s	6018	55m43.29s	5954	82m56.78s	5992	796m39.90s	**5925**	1104m52.68s	7962	35m43.7s
G900	7244	102m36.3s	7254	76m44.54s	7346	131m19.5s	7244	1556m46.53s	**7238**	1864m28.23s	9488	45m42.96
G1000	8206	124m2.26s	8245	78m51.23s	8224	214m50.6s	8219	1282m23.44s	**8201**	2822m04.63s	10719	56m50.13

Tabela 5- Resultados obtidos com a aplicação dos algoritmos baseados no GRASP e VNS comparados ao H1

Em relação ao custo das soluções, os melhores resultados foram obtidos com o GRASP4, enquanto que, o GRASP2 obteve os piores custos. Para as instâncias com mais de 400 nós, o tempo de execução do GRASP4 foi maior. A própria definição do GRASP3 e GRASP4 estabelece um tempo de execução maior quando são comparados aos outros algoritmos, uma vez que, no GRASP3 e GRASP4, além da instância original, as variações também são consideradas. Por outro lado, ao comparar o tempo de execução do GRASP3 com GRASP4, deve-se considerar que o método de busca definido no GRASP4 é o VNSP que, de acordo com a instância, pode exigir um tempo computacional maior que o USM, definido na fase de busca do GRASP3.

Para todas as instâncias geradas, pelo menos um dos algoritmos propostos obteve soluções melhores que H1.

O GRASP3 não obteve melhores soluções que H1 apenas nas instâncias G500 e G800. Já o VNS1 obteve soluções piores que H1 somente nas instâncias G200 e G800. Além disso, o VNS2 quando comparado ao H1, não obteve melhores soluções somente nas instâncias G500, G800 e G1000.

Tanto o VNS1, quanto o VNS2, exigiram um tempo de execução inferior ao de H1. Uma justificativa para esta diferença entre o tempo computacional desses dois algoritmos quando comparados ao H1, se deve à maneira como GENI e US são aplicados nas fases de construção e busca local.

Na fase de construção do H1, o tamanho da vizinhança considerada pelo GENI é igual a 4, enquanto que, no VNS1 e VNS2, p = 6. Portanto as soluções iniciais obtidas por esses algoritmos são melhores que no H1.

Em relação à fase de busca, a própria definição dos métodos de busca do H1 e do VNS1 determinam uma diferença entre o tempo computacional dos dois métodos. Além disso, US é realizado com p = 4 durante toda a fase de busca local e nos algoritmos VNS2 e VNS1, a busca começa com p = 2 e cresce a medida que o algoritmo evolui. Porem, o número de iterações com p > 2, realizadas no VNS2 e VNS1, é menor que no H1, pois quando p = 4, no VNS2 e VNS1, o algoritmo não consegue obter muita melhora, visto que, a solução inicial é melhor nesses dois algoritmos.

Quando o VNS1 é comparado ao VNS2, o VNS2 obteve melhores soluções nas instâncias G200 e G400. No G50, G100 e G600 as soluções obtidas pelos dois algoritmos foram iguais. Nas demais instâncias, o VNS1 obteve melhores soluções que VNS2.

Em relação ao GRASP2, o algoritmo obteve soluções piores que H1 em todas as instâncias. Porém, nas instâncias com mais de 500 nos, GRASP2 obteve um tempo de execução menor que de H1.

Finalmente, na Tabela 6, é apresentada a média dos resultados obtidos com os testes realizados aplicando os algoritmos GRAS3, GRASP4, VNS1, VNS2 e H1, variando o número de grupamentos das instâncias. Os algoritmos VNS1, VNS2 e H1 foram executados 10 vezes em cada instância. Já os algoritmos GRASP3 e GRASP4 foram executados uma vez.

Foram realizados testes nas instâncias G300, G400 e G500 e o número de grupamentos foram iguais a 5%, 10% e 20% do número de vértices. Os testes foram realizados em um PC com processador Pentium III, 650 MHz, 250 Mb de memória, sistema operacional Linux.

Grafo	M	GRASP3		GRASP4		H1		VNS1		VNS2	
		C	T	C	T	C	T	C	T	C	T
G300	**15**	1461	30m38.50s	1445	28m16.75s	1440	1m59.15s	**1421**	1m9.19s	**1421**	1m17.90s
	30	2467	30m24.97s	**2444**	23m52.74s	2515	1m15.01s	2473	1m7.88s	2481	1m3.06s
	45	3015	34m32.56s	**2985**	28m0.01s	3159	1m52.28s	3069	1m2.60s	3040	1m28.71s
G400	**20**	2020	74m36.23s	**2010**	74m20.77s	2023	3m58.54s	2018	3m36.14s	2028	2m55.95s
	40	**2985**	74m55.70s	3000	69m2.83s	3027	4m45.19s	3013	2m26.55s	2996	3m3.3s
	80	**4015**	121m40.15s	4124	90m33.76s	4260	9m10.31s	4259	3m2.86s	4322	2m33.97s
G500	**25**	2452	150m38.75s	**2428**	178m19.80s	2434	8m27.48s	2457	5m37.11s	2437	7m31.84s
	50	4108	154m27.61s	**4089**	169m32.32s	4101	10m27.37s	4093	11m24.49s	4127	7m52.14s
	100	5468	317m41.60s	**5449**	252m38.38s	5601	20m32.11s	5522	9m12.33s	5484	14m1.72s

Tabela 6 – Resultados obtidos com a aplicação dos algoritmos variando o número de grupamentos da instância

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G50	407	0.30s	413	0.31s	541	0.02s	479	0.04s	530	0.012s
V1	398	0.30s	398	0.31s	477	0.02s	578	0.04s	533	0.012s
V2	383	0.30s	383	0.31s	491	0.02s	516	0.04s	503	0.012s
V3	397	0.30s	397	0.31s	523	0.02s	546	0.04s	493	0.012s
V4	404	0.30s	404	0.31s	515	0.02s	531	0.04s	545	0.012s
V5	389	0.30s	388	0.31s	504	0.02s	583	0.04s	**455**	0.012s
V6	416	0.30s	416	0.31s	449	0.02s	527	0.04s	513	0.012s
V7	383	0.30s	383	0.31s	525	0.02s	483	0.04s	525	0.012s
V8	390	0.30s	390	0.31s	560	0.02s	568	0.04s	577	0.012s
V9	401	0.30s	401	0.31s	530	0.02s	506	0.04s	550	0.012s
V10	395	0.30s	403	0.31s	502	0.02s	497	0.04s	549	0.012s
V11	395	0.30s	395	0.31s	479	0.02s	512	0.04s	523	0.012s
V12	396	0.30s	383	0.31s	557	0.02s	562	0.04s	588	0.012s
V13	**368**	0.30s	**377**	0.31s	499	0.02s	510	0.04s	530	0.012s
V14	390	0.30s	396	0.31s	511	0.02s	505	0.04s	501	0.012s
V15	410	0.30s	404	0.31s	523	0.02s	**474**	0.04s	477	0.012s
V16	393	0.30s	393	0.31s	514	0.02s	581	0.04s	529	0.012s
V17	381	0.30s	381	0.31s	449	0.02s	515	0.04s	511	0.012s
V18	382	0.30s	394	0.31s	531	0.02s	543	0.04s	619	0.012s
V19	401	0.30s	411	0.31s	525	0.02s	511	0.04s	539	0.012s
V20	397	0.30s	398	0.31s	563	0.02s	521	0.04s	538	0.012s
V21	396	0.30s	389	0.31s	542	0.02s	506	0.04s	497	0.012s
V22	399	0.30s	392	0.31s	551	0.02s	520	0.04s	570	0.012s
V23	383	0.30s	**377**	0.31s	473	0.02s	520	0.04s	574	0.012s
V24	397	0.30s	398	0.31s	505	0.02s	509	0.04s	482	0.012s
V25	396	0.30s	391	0.31s	544	0.02s	527	0.04s	566	0.012s
V26	397	0.30s	403	0.31s	532	0.02s	544	0.04s	523	0.012s
V27	398	0.30s	384	0.31s	514	0.02s	574	0.04s	547	0.012s
V28	405	0.30s	405	0.31s	498	0.02s	556	0.04s	518	0.012s
V29	399	0.30s	399	0.31s	498	0.02s	528	0.04s	597	0.012s
V30	422	0.30s	423	0.31s	520	0.02s	524	0.04s	539	0.012s
V31	404	0.30s	404	0.31s	522	0.02s	518	0.04s	573	0.012s
V32	402	0.30s	399	0.31s	512	0.02s	571	0.04s	572	0.012s
V33	422	0.30s	425	0.31s	534	0.02s	589	0.04s	575	0.012s
V34	413	0.30s	407	0.31s	506	0.02s	568	0.04s	586	0.012s
V35	403	0.30s	406	0.31s	587	0.02s	553	0.04s	592	0.012s
V36	393	0.30s	395	0.31s	486	0.02s	492	0.04s	483	0.012s
V37	385	0.30s	385	0.31s	524	0.02s	517	0.04s	524	0.012s
V38	410	0.30s	390	0.31s	482	0.02s	558	0.04s	583	0.012s
V39	392	0.30s	398	0.31s	520	0.02s	533	0.04s	545	0.012s
V40	396	0.30s	390	0.31s	494	0.02s	546	0.04s	513	0.012s
V41	393	0.30s	383	0.31s	535	0.02s	488	0.04s	536	0.012s
V42	396	0.30s	406	0.31s	496	0.02s	566	0.04s	510	0.012s
V43	386	0.30s	403	0.31s	507	0.02s	560	0.04s	571	0.012s
V44	406	0.30s	422	0.31s	529	0.02s	494	0.04s	511	0.012s
V45	420	0.30s	421	0.31s	474	0.02s	524	0.04s	538	0.012s
V46	390	0.30s	388	0.31s	580	0.02s	538	0.04s	476	0.012s
V47	389	0.30s	**377**	0.31s	541	0.02s	538	0.04s	529	0.012s
V48	396	0.30s	396	0.31s	523	0.02s	508	0.04s	492	0.012s
V49	401	0.30s	393	0.31s	514	0.02s	484	0.04s	577	0.012s
V50	388	0.30s	388	0.31s	**446**	0.02s	512	0.04s	515	0.012s

Tabela 7 - Resultados para o G50

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G100	772	1.549s	760	2.332s	1034	0.214s	1012	0.368s	900	0.036s
V1	759	1.549s	773	2.332s	1060	0.214s	965	0.368s	929	0.036s
V2	763	1.549s	741	2.332s	1044	0.214s	984	0.368s	998	0.036s
V3	799	1.549s	789	2.332s	1083	0.214s	1099	0.368s	963	0.036s
V4	791	1.549s	795	2.332s	1081	0.214s	1083	0.368s	929	0.036s
V5	779	1.549s	772	2.332s	1015	0.214s	1047	0.368s	**819**	0.036s
V6	773	1.549s	771	2.332s	1080	0.214s	1013	0.368s	912	0.036s
V7	787	1.549s	772	2.332s	1076	0.214s	1119	0.368s	944	0.036s
V8	771	1.549s	749	2.332s	1100	0.214s	1054	0.368s	1035	0.036s
V9	**741**	1.549s	748	2.332s	1030	0.214s	1038	0.368s	878	0.036s
V10	788	1.549s	786	2.332s	1066	0.214s	1153	0.368s	938	0.036s
V11	767	1.549s	755	2.332s	1123	0.214s	1007	0.368s	972	0.036s
V12	752	1.549s	774	2.332s	1076	0.214s	987	0.368s	897	0.036s
V13	783	1.549s	801	2.332s	1019	0.214s	977	0.368s	937	0.036s
V14	748	1.549s	756	2.332s	1140	0.214s	1007	0.368s	923	0.036s
V15	772	1.549s	758	2.332s	1052	0.214s	1059	0.368s	920	0.036s
V16	789	1.549s	789	2.332s	1060	0.214s	1047	0.368s	935	0.036s
V17	765	1.549s	758	2.332s	1035	0.214s	1005	0.368s	854	0.036s
V18	786	1.549s	741	2.332s	1146	0.214s	965	0.368s	929	0.036s
V19	744	1.549s	**737**	2.332s	1062	0.214s	1093	0.368s	911	0.036s
V20	772	1.549s	769	2.332s	1060	0.214s	977	0.368s	920	0.036s
V21	810	1.549s	810	2.332s	1095	0.214s	1003	0.368s	916	0.036s
V22	780	1.549s	780	2.332s	1074	0.214s	1113	0.368s	925	0.036s
V23	790	1.549s	812	2.332s	1029	0.214s	1034	0.368s	996	0.036s
V24	780	1.549s	794	2.332s	989	0.214s	978	0.368s	951	0.036s
V25	770	1.549s	753	2.332s	1075	0.214s	1063	0.368s	962	0.036s
V26	768	1.549s	770	2.332s	1167	0.214s	1196	0.368s	965	0.036s
V27	767	1.549s	763	2.332s	1006	0.214s	1016	0.368s	887	0.036s
V28	766	1.549s	791	2.332s	1141	0.214s	990	0.368s	918	0.036s
V29	758	1.549s	780	2.332s	1139	0.214s	1103	0.368s	958	0.036s
V30	755	1.549s	765	2.332s	1037	0.214s	1018	0.368s	1005	0.036s
V31	783	1.549s	788	2.332s	1086	0.214s	1080	0.368s	987	0.036s
V32	771	1.549s	771	2.332s	1053	0.214s	1028	0.368s	946	0.036s
V33	798	1.549s	810	2.332s	1069	0.214s	1002	0.368s	994	0.036s
V34	810	1.549s	806	2.332s	1110	0.214s	1082	0.368s	894	0.036s
V35	774	1.549s	774	2.332s	1052	0.214s	1016	0.368s	936	0.036s
V36	783	1.549s	797	2.332s	**988**	0.214s	1013	0.368s	943	0.036s
V37	787	1.549s	798	2.332s	1097	0.214s	986	0.368s	921	0.036s
V38	764	1.549s	787	2.332s	1067	0.214s	966	0.368s	836	0.036s
V39	766	1.549s	752	2.332s	1125	0.214s	1056	0.368s	983	0.036s
V40	772	1.549s	760	2.332s	1086	0.214s	1061	0.368s	977	0.036s
V41	780	1.549s	763	2.332s	998	0.214s	**901**	0.368s	910	0.036s
V42	772	1.549s	739	2.332s	1068	0.214s	1038	0.368s	965	0.036s
V43	754	1.549s	754	2.332s	1041	0.214s	1025	0.368s	925	0.036s
V44	749	1.549s	769	2.332s	1162	0.214s	1058	0.368s	922	0.036s
V45	775	1.549s	766	2.332s	1086	0.214s	1066	0.368s	973	0.036s
V46	754	1.549s	757	2.332s	1129	0.214s	1036	0.368s	930	0.036s
V47	770	1.549s	783	2.332s	1151	0.214s	1081	0.368s	982	0.036s
V48	759	1.549s	761	2.332s	1027	0.214s	989	0.368s	1043	0.036s
V49	775	1.549s	783	2.332s	1030	0.214s	1039	0.368s	930	0.036s
V50	776	1.549s	788	2.332s	1086	0.214s	1025	0.368s	983	0.036s

Tabela 8 - Resultados para o G100

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G200	1648	10.5s	1631	22.61s	2231	3.288s	2081	5.320s	2039	0.185s
V1	1681	10.5s	1665	22.61s	2291	3.288s	2162	5.320s	2026	0.185s
V2	1690	10.5s	1661	22.61s	2243	3.288s	2168	5.320s	1958	0.185s
V3	1659	10.5s	1650	22.61s	2209	3.288s	2069	5.320s	2060	0.185s
V4	1726	10.5s	1676	22.61s	2233	3.288s	2109	5.320s	2060	0.185s
V5	1684	10.5s	1659	22.61s	2172	3.288s	2150	5.320s	2049	0.185s
V6	1739	10.5s	1678	22.61s	2138	3.288s	2202	5.320s	1969	0.185s
V7	1669	10.5s	1754	22.61s	2169	3.288s	2205	5.320s	2154	0.185s
V8	1666	10.5s	1651	22.61s	**2087**	3.288s	2110	5.320s	2052	0.185s
V9	1668	10.5s	1644	22.61s	2234	3.288s	2027	5.320s	2020	0.185s
V10	1629	10.5s	1626	22.61s	2224	3.288s	2191	5.320s	2051	0.185s
V11	1663	10.5s	**1611**	22.61s	2232	3.288s	2257	5.320s	2073	0.185s
V12	1710	10.5s	1765	22.61s	2168	3.288s	2090	5.320s	2169	0.185s
V13	1662	10.5s	1656	22.61s	2286	3.288s	2119	5.320s	2113	0.185s
V14	1670	10.5s	1653	22.61s	2223	3.288s	2177	5.320s	1996	0.185s
V15	1651	10.5s	1638	22.61s	2189	3.288s	2241	5.320s	2045	0.185s
V16	1628	10.5s	1637	22.61s	2235	3.288s	1979	5.320s	2143	0.185s
V17	1693	10.5s	1712	22.61s	2317	3.288s	2066	5.320s	2031	0.185s
V18	1633	10.5s	1636	22.61s	2334	3.288s	2267	5.320s	2059	0.185s
V19	1663	10.5s	1659	22.61s	2153	3.288s	2077	5.320s	2019	0.185s
V20	1754	10.5s	1760	22.61s	2259	3.288s	2180	5.320s	1953	0.185s
V21	1684	10.5s	1680	22.61s	2271	3.288s	2130	5.320s	2006	0.185s
V22	1630	10.5s	1696	22.61s	2282	3.288s	2146	5.320s	2027	0.185s
V23	**1605**	10.5s	1616	22.61s	2335	3.288s	2085	5.320s	2047	0.185s
V24	1724	10.5s	1688	22.61s	2210	3.288s	2130	5.320s	1975	0.185s
V25	1663	10.5s	1667	22.61s	2128	3.288s	2040	5.320s	2054	0.185s
V26	1680	10.5s	1657	22.61s	2181	3.288s	2068	5.320s	2084	0.185s
V27	1662	10.5s	1654	22.61s	2321	3.288s	2137	5.320s	2082	0.185s
V28	1701	10.5s	1698	22.61s	2142	3.288s	2262	5.320s	2001	0.185s
V29	1646	10.5s	1667	22.61s	2312	3.288s	2131	5.320s	1942	0.185s
V30	1620	10.5s	1700	22.61s	2201	3.288s	2179	5.320s	**1900**	0.185s
V31	1712	10.5s	1701	22.61s	2144	3.288s	2135	5.320s	2137	0.185s
V32	1713	10.5s	1635	22.61s	2300	3.288s	2199	5.320s	1975	0.185s
V33	1707	10.5s	1682	22.61s	2098	3.288s	2183	5.320s	2033	0.185s
V34	1677	10.5s	1658	22.61s	2220	3.288s	2186	5.320s	2055	0.185s
V35	1681	10.5s	1691	22.61s	2223	3.288s	2151	5.320s	2019	0.185s
V36	1690	10.5s	1683	22.61s	2088	3.288s	**2005**	5.320s	2015	0.185s
V37	1670	10.5s	1658	22.61s	2185	3.288s	2152	5.320s	2000	0.185s
V38	1647	10.5s	1638	22.61s	2180	3.288s	2295	5.320s	2008	0.185s
V39	1678	10.5s	1696	22.61s	2282	3.288s	2146	5.320s	2115	0.185s
V40	1618	10.5s	1629	22.61s	2246	3.288s	2022	5.320s	2103	0.185s
V41	1698	10.5s	1720	22.61s	2219	3.288s	2164	5.320s	2117	0.185s
V42	1648	10.5s	1643	22.61s	2152	3.288s	2167	5.320s	1978	0.185s
V43	1669	10.5s	1709	22.61s	2239	3.288s	2027	5.320s	2092	0.185s
V44	1673	10.5s	1708	22.61s	2247	3.288s	2113	5.320s	2128	0.185s
V45	1672	10.5s	1613	22.61s	2214	3.288s	2210	5.320s	2038	0.185s
V46	1711	10.5s	1697	22.61s	2221	3.288s	2042	5.320s	2072	0.185s
V47	1676	10.5s	1645	22.61s	2306	3.288s	2262	5.320s	2023	0.185s
V48	1643	10.5s	1642	22.61s	2282	3.288s	2139	5.320s	2020	0.185s
V49	1686	10.5s	1668	22.61s	2219	3.288s	2276	5.320s	1972	0.185s
V50	1702	10.5s	1668	22.61s	2235	3.288s	2181	5.320s	2029	0.185s

Tabela 9 - Resultados para o G200

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G300	2510	35.388s	2515	1m45.59s	3228	20.48s	3281	28.29s	2890	0.572s
V1	2512	35.388s	2515	1m45.59s	3410	20.48s	3194	28.29s	2981	0.572s
V2	2496	35.388s	2494	1m45.59s	3272	20.48s	3199	28.29s	3067	0.572s
V3	2535	35.388s	2531	1m45.59s	3316	20.48s	3187	28.29s	2954	0.572s
V4	2579	35.388s	2522	1m45.59s	3182	20.48s	3224	28.29s	3033	0.572s
V5	2560	35.388s	2521	1m45.59s	3250	20.48s	3298	28.29s	3030	0.572s
V6	2596	35.388s	2596	1m45.59s	3298	20.48s	3322	28.29s	2963	0.572s
V7	2578	35.388s	2521	1m45.59s	3366	20.48s	3396	28.29s	2995	0.572s
V8	2585	35.388s	2579	1m45.59s	3204	20.48s	3175	28.29s	3139	0.572s
V9	2536	35.388s	2528	1m45.59s	3242	20.48s	3276	28.29s	3004	0.572s
V10	2567	35.388s	2518	1m45.59s	3280	20.48s	**3000**	28.29s	3023	0.572s
V11	2551	35.388s	2512	1m45.59s	3220	20.48s	3470	28.29s	2998	0.572s
V12	2554	35.388s	2562	1m45.59s	3296	20.48s	3168	28.29s	3005	0.572s
V13	2581	35.388s	2595	1m45.59s	3374	20.48s	3276	28.29s	3016	0.572s
V14	2588	35.388s	2562	1m45.59s	3389	20.48s	3256	28.29s	3079	0.572s
V15	2509	35.388s	2515	1m45.59s	3385	20.48s	3089	28.29s	2978	0.572s
V16	2551	35.388s	2492	1m45.59s	3222	20.48s	3326	28.29s	2935	0.572s
V17	2585	35.388s	2539	1m45.59s	3333	20.48s	3248	28.29s	2973	0.572s
V18	2516	35.388s	2538	1m45.59s	3386	20.48s	3303	28.29s	3006	0.572s
V19	2508	35.388s	**2447**	1m45.59s	3336	20.48s	3271	28.29s	2914	0.572s
V20	2519	35.388s	2470	1m45.59s	3195	20.48s	3203	28.29s	3007	0.572s
V21	**2481**	35.388s	2478	1m45.59s	3256	20.48s	3229	28.29s	2996	0.572s
V22	2538	35.388s	2508	1m45.59s	3256	20.48s	3235	28.29s	2925	0.572s
V23	2560	35.388s	2565	1m45.59s	3247	20.48s	3345	28.29s	2994	0.572s
V24	2513	35.388s	2500	1m45.59s	3342	20.48s	3177	28.29s	2983	0.572s
V25	2559	35.388s	2572	1m45.59s	3340	20.48s	3155	28.29s	3099	0.572s
V26	2504	35.388s	2569	1m45.59s	3113	20.48s	3211	28.29s	3015	0.572s
V27	2591	35.388s	2602	1m45.59s	3214	20.48s	3163	28.29s	3069	0.572s
V28	2555	35.388s	2531	1m45.59s	3331	20.48s	3160	28.29s	3170	0.572s
V29	2535	35.388s	2561	1m45.59s	3218	20.48s	3171	28.29s	3126	0.572s
V30	2515	35.388s	2514	1m45.59s	3275	20.48s	3309	28.29s	3031	0.572s
V31	2528	35.388s	2525	1m45.59s	3211	20.48s	3306	28.29s	**2884**	0.572s
V32	2542	35.388s	2601	1m45.59s	3281	20.48s	3314	28.29s	3070	0.572s
V33	2553	35.388s	2584	1m45.59s	3322	20.48s	3380	28.29s	3053	0.572s
V34	2602	35.388s	2577	1m45.59s	3298	20.48s	3184	28.29s	2975	0.572s
V35	2557	35.388s	2495	1m45.59s	3268	20.48s	3216	28.29s	3090	0.572s
V36	2498	35.388s	2509	1m45.59s	3386	20.48s	3285	28.29s	3082	0.572s
V37	2483	35.388s	2487	1m45.59s	3228	20.48s	3279	28.29s	2970	0.572s
V38	2565	35.388s	2545	1m45.59s	3202	20.48s	3311	28.29s	2999	0.572s
V39	2549	35.388s	2547	1m45.59s	3316	20.48s	3316	28.29s	3056	0.572s
V40	2552	35.388s	2604	1m45.59s	3224	20.48s	3232	28.29s	3095	0.572s
V41	2562	35.388s	2562	1m45.59s	**3081**	20.48s	3369	28.29s	2956	0.572s
V42	2559	35.388s	2613	1m45.59s	3269	20.48s	3318	28.29s	2911	0.572s
V43	2545	35.388s	2536	1m45.59s	3256	20.48s	3258	28.29s	2965	0.572s
V44	2558	35.388s	2562	1m45.59s	3343	20.48s	3205	28.29s	2983	0.572s
V45	2502	35.388s	2504	1m45.59s	3278	20.48s	3234	28.29s	2942	0.572s
V46	2579	35.388s	2580	1m45.59s	3235	20.48s	3165	28.29s	2974	0.572s
V47	2572	35.388s	2521	1m45.59s	3392	20.48s	3167	28.29s	2978	0.572s
V48	2483	35.388s	2483	1m45.59s	3165	20.48s	3249	28.29s	3043	0.572s
V49	2565	35.388s	2499	1m45.59s	3366	20.48s	3308	28.29s	3035	0.572s
V50	2574	35.388s	2604	1m45.59s	3239	20.48s	3281	28.29s	2985	0.572s

Tabela 10- Resultados para o G300

Grafo	HiM		Hi		HiMP		HiMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G400	3048	1m18.08s	3027	4m7.04s	4135	1m17.55s	3913	1m37.94s	3744	1.49s
V1	3105	1m18.08s	3069	4m7.04s	4282	1m17.55s	3902	1m37.94s	3844	1.49s
V2	3038	1m18.08s	3040	4m7.04s	4421	1m17.55s	4033	1m37.94s	3970	1.49s
V3	3105	1m18.08s	3145	4m7.04s	4198	1m17.55s	4059	1m37.94s	3906	1.49s
V4	3109	1m18.08s	3070	4m7.04s	4191	1m17.55s	3952	1m37.94s	3805	1.49s
V5	3070	1m18.08s	3048	4m7.04s	4251	1m17.55s	4008	1m37.94s	3999	1.49s
V6	3126	1m18.08s	3131	4m7.04s	4161	1m17.55s	4195	1m37.94s	3834	1.49s
V7	3105	1m18.08s	3119	4m7.04s	4243	1m17.55s	4078	1m37.94s	3957	1.49s
V8	3136	1m18.08s	3112	4m7.04s	4301	1m17.55s	4011	1m37.94s	3988	1.49s
V9	3081	1m18.08s	3049	4m7.04s	4268	1m17.55s	4164	1m37.94s	3903	1.49s
V10	3104	1m18.08s	3091	4m7.04s	4126	1m17.55s	4107	1m37.94s	3793	1.49s
V11	3198	1m18.08s	3226	4m7.04s	4120	1m17.55s	4124	1m37.94s	3854	1.49s
V12	3108	1m18.08s	3123	4m7.04s	4295	1m17.55s	4088	1m37.94s	3834	1.49s
V13	3115	1m18.08s	3121	4m7.04s	4193	1m17.55s	3918	1m37.94s	3833	1.49s
V14	3153	1m18.08s	3136	4m7.04s	4206	1m17.55s	3997	1m37.94s	3875	1.49s
V15	3071	1m18.08s	3144	4m7.04s	4135	1m17.55s	4017	1m37.94s	3817	1.49s
V16	3085	1m18.08s	3076	4m7.04s	4199	1m17.55s	4003	1m37.94s	3958	1.49s
V17	3098	1m18.08s	3113	4m7.04s	4314	1m17.55s	3941	1m37.94s	3858	1.49s
V18	3067	1m18.08s	3146	4m7.04s	4316	1m17.55s	4024	1m37.94s	3906	1.49s
V19	3178	1m18.08s	3154	4m7.04s	4323	1m17.55s	4072	1m37.94s	3898	1.49s
V20	3104	1m18.08s	3094	4m7.04s	4200	1m17.55s	4055	1m37.94s	3739	1.49s
V21	3095	1m18.08s	3085	4m7.04s	4201	1m17.55s	4042	1m37.94s	3680	1.49s
V22	3186	1m18.08s	3143	4m7.04s	4141	1m17.55s	3954	1m37.94s	3935	1.49s
V23	3086	1m18.08s	3148	4m7.04s	4206	1m17.55s	4078	1m37.94s	3898	1.49s
V24	**3023**	1m18.08s	3055	4m7.04s	**3085**	1m17.55s	4028	1m37.94s	3732	1.49s
V25	3176	1m18.08s	3163	4m7.04s	4266	1m17.55s	4086	1m37.94s	3740	1.49s
V26	3123	1m18.08s	3085	4m7.04s	4293	1m17.55s	3968	1m37.94s	3954	1.49s
V27	3108	1m18.08s	3133	4m7.04s	4263	1m17.55s	4027	1m37.94s	3729	1.49s
V28	3151	1m18.08s	3151	4m7.04s	4259	1m17.55s	4044	1m37.94s	3985	1.49s
V29	3077	1m18.08s	3111	4m7.04s	4078	1m17.55s	4107	1m37.94s	3953	1.49s
V30	3141	1m18.08s	3122	4m7.04s	4334	1m17.55s	4033	1m37.94s	3863	1.49s
V31	3129	1m18.08s	3137	4m7.04s	4310	1m17.55s	4041	1m37.94s	3727	1.49s
V32	3059	1m18.08s	3039	4m7.04s	4055	1m17.55s	4102	1m37.94s	3829	1.49s
V33	3062	1m18.08s	**3025**	4m7.04s	4162	1m17.55s	3963	1m37.94s	3727	1.49s
V34	3088	1m18.08s	3101	4m7.04s	4124	1m17.55s	3899	1m37.94s	3901	1.49s
V35	3118	1m18.08s	3113	4m7.04s	4051	1m17.55s	4119	1m37.94s	3790	1.49s
V36	3093	1m18.08s	3089	4m7.04s	4152	1m17.55s	4016	1m37.94s	3797	1.49s
V37	3111	1m18.08s	3140	4m7.04s	4206	1m17.55s	3976	1m37.94s	4006	1.49s
V38	3108	1m18.08s	3119	4m7.04s	3943	1m17.55s	3863	1m37.94s	**3616**	1.49s
V39	3089	1m18.08s	3090	4m7.04s	4337	1m17.55s	4051	1m37.94s	3773	1.49s
V40	3066	1m18.08s	3049	4m7.04s	4083	1m17.55s	4045	1m37.94s	3842	1.49s
V41	3094	1m18.08s	3097	4m7.04s	4178	1m17.55s	3911	1m37.94s	3779	1.49s
V42	3094	1m18.08s	3067	4m7.04s	4206	1m17.55s	4114	1m37.94s	3805	1.49s
V43	3098	1m18.08s	3122	4m7.04s	4170	1m17.55s	4146	1m37.94s	3844	1.49s
V44	3131	1m18.08s	3101	4m7.04s	4037	1m17.55s	4015	1m37.94s	3892	1.49s
V45	3086	1m18.08s	3138	4m7.04s	4071	1m17.55s	3978	1m37.94s	3778	1.49s
V46	3097	1m18.08s	3111	4m7.04s	4235	1m17.55s	4137	1m37.94s	3812	1.49s
V47	3109	1m18.08s	3041	4m7.04s	4141	1m17.55s	4060	1m37.94s	3791	1.49s
V48	3109	1m18.08s	3095	4m7.04s	4115	1m17.55s	3956	1m37.94s	3737	1.49s
V49	3070	1m18.08s	3072	4m7.04s	4048	1m17.55s	**3879**	1m37.94s	3832	1.49s
V50	3093	1m18.08s	3053	4m7.04s	4273	1m17.55s	3972	1m37.94s	4060	1.49s

Tabela 11 - Resultados para o G400

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G500	**4091**	2m48s	**4101**	10m46.16s	5533	3m43.29s	5543	4m40.17s	5022	3.21s
V1	4187	2m48s	4187	10m46.16s	5607	3m43.29s	5561	4m40.17s	5108	3.21s
V2	4255	2m48s	4207	10m46.16s	5665	3m43.29s	5493	4m40.17s	5076	3.21s
V3	4283	2m48s	4202	10m46.16s	5560	3m43.29s	5372	4m40.17s	5138	3.21s
V4	4210	2m48s	4235	10m46.16s	5540	3m43.29s	5436	4m40.17s	5114	3.21s
V5	4152	2m48s	4179	10m46.16s	5586	3m43.29s	5305	4m40.17s	5323	3.21s
V6	4204	2m48s	4172	10m46.16s	5661	3m43.29s	5380	4m40.17s	5186	3.21s
V7	4262	2m48s	4213	10m46.16s	5516	3m43.29s	5442	4m40.17s	5043	3.21s
V8	4229	2m48s	4199	10m46.16s	5548	3m43.29s	5361	4m40.17s	5005	3.21s
V9	4208	2m48s	4186	10m46.16s	5552	3m43.29s	5240	4m40.17s	5224	3.21s
V10	4153	2m48s	4214	10m46.16s	5760	3m43.29s	5461	4m40.17s	**4882**	3.21s
V11	4222	2m48s	4220	10m46.16s	5608	3m43.29s	5243	4m40.17s	4969	3.21s
V12	4184	2m48s	4143	10m46.16s	5605	3m43.29s	5311	4m40.17s	5130	3.21s
V13	4251	2m48s	4262	10m46.16s	5416	3m43.29s	5369	4m40.17s	5234	3.21s
V14	4287	2m48s	4287	10m46.16s	5582	3m43.29s	5272	4m40.17s	5002	3.21s
V15	4253	2m48s	4235	10m46.16s	5489	3m43.29s	5337	4m40.17s	5019	3.21s
V16	4181	2m48s	4169	10m46.16s	5485	3m43.29s	5495	4m40.17s	4947	3.21s
V17	4153	2m48s	4177	10m46.16s	5622	3m43.29s	5488	4m40.17s	5235	3.21s
V18	4120	2m48s	4118	10m46.16s	5597	3m43.29s	5520	4m40.17s	4977	3.21s
V19	4167	2m48s	4185	10m46.16s	5525	3m43.29s	5476	4m40.17s	5063	3.21s
V20	4153	2m48s	4181	10m46.16s	5705	3m43.29s	5174	4m40.17s	5172	3.21s
V21	4210	2m48s	4194	10m46.16s	5589	3m43.29s	5456	4m40.17s	5051	3.21s
V22	4254	2m48s	4229	10m46.16s	5632	3m43.29s	5382	4m40.17s	5070	3.21s
V23	4155	2m48s	4138	10m46.16s	5670	3m43.29s	5393	4m40.17s	5132	3.21s
V24	4205	2m48s	4246	10m46.16s	5611	3m43.29s	5322	4m40.17s	5066	3.21s
V25	4236	2m48s	4187	10m46.16s	5648	3m43.29s	5299	4m40.17s	5037	3.21s
V26	4145	2m48s	4192	10m46.16s	5653	3m43.29s	5418	4m40.17s	5000	3.21s
V27	4233	2m48s	4281	10m46.16s	5432	3m43.29s	5404	4m40.17s	5233	3.21s
V28	4212	2m48s	4277	10m46.16s	5438	3m43.29s	5440	4m40.17s	4884	3.21s
V29	4168	2m48s	4148	10m46.16s	5494	3m43.29s	5367	4m40.17s	5095	3.21s
V30	4308	2m48s	4254	10m46.16s	5590	3m43.29s	5412	4m40.17s	5077	3.21s
V31	4211	2m48s	4195	10m46.16s	5508	3m43.29s	5466	4m40.17s	5019	3.21s
V32	4247	2m48s	4216	10m46.16s	5662	3m43.29s	5295	4m40.17s	5168	3.21s
V33	4193	2m48s	4171	10m46.16s	5740	3m43.29s	5346	4m40.17s	5142	3.21s
V34	4151	2m48s	4142	10m46.16s	5567	3m43.29s	5319	4m40.17s	5088	3.21s
V35	4184	2m48s	4209	10m46.16s	5581	3m43.29s	5313	4m40.17s	5042	3.21s
V36	4255	2m48s	4246	10m46.16s	5758	3m43.29s	**5146**	4m40.17s	5198	3.21s
V37	4153	2m48s	4133	10m46.16s	5580	3m43.29s	5274	4m40.17s	5033	3.21s
V38	4235	2m48s	4259	10m46.16s	**5403**	3m43.29s	5294	4m40.17s	5126	3.21s
V39	4161	2m48s	4226	10m46.16s	5440	3m43.29s	5720	4m40.17s	5068	3.21s
V40	4229	2m48s	4273	10m46.16s	5587	3m43.29s	5342	4m40.17s	4952	3.21s
V41	4210	2m48s	4210	10m46.16s	5530	3m43.29s	5379	4m40.17s	5046	3.21s
V42	4189	2m48s	4193	10m46.16s	5487	3m43.29s	5564	4m40.17s	5046	3.21s
V43	4267	2m48s	4304	10m46.16s	5554	3m43.29s	5268	4m40.17s	4991	3.21s
V44	4231	2m48s	4155	10m46.16s	5621	3m43.29s	5422	4m40.17s	5221	3.21s
V45	4223	2m48s	4189	10m46.16s	5625	3m43.29s	5421	4m40.17s	5147	3.21s
V46	4223	2m48s	4267	10m46.16s	5653	3m43.29s	5529	4m40.17s	4976	3.21s
V47	4203	2m48s	4198	10m46.16s	5696	3m43.29s	5513	4m40.17s	5214	3.21s
V48	4199	2m48s	4159	10m46.16s	5385	3m43.29s	5355	4m40.17s	5329	3.21s
V49	4247	2m48s	4275	10m46.16s	5563	3m43.29s	5626	4m40.17s	5004	3.21s
V50	4197	2m48s	4195	10m46.16s	5626	3m43.29s	5284	4m40.17s	4940	3.21s

Tabela 12 - Resultados para o G500

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G600	**4252**	4m58.545s	**4263**	20m18.42s	**5758**	8m19.78s	5914	10m31.23s	**5223**	6.3s
V1	4496	4m58.545s	4454	20m18.42s	5995	8m19.78s	6045	10m31.23s	5272	6.3s
V2	4350	4m58.545s	4366	20m18.42s	6020	8m19.78s	5737	10m31.23s	5419	6.3s
V3	4392	4m58.545s	4381	20m18.42s	6128	8m19.78s	5777	10m31.23s	5521	6.3s
V4	4311	4m58.545s	4325	20m18.42s	6018	8m19.78s	**5672**	10m31.23s	5649	6.3s
V5	4300	4m58.545s	4332	20m18.42s	6040	8m19.78s	5799	10m31.23s	5541	6.3s
V6	4414	4m58.545s	4456	20m18.42s	6065	8m19.78s	5859	10m31.23s	5351	6.3s
V7	4373	4m58.545s	4366	20m18.42s	5972	8m19.78s	5854	10m31.23s	5301	6.3s
V8	4455	4m58.545s	4414	20m18.42s	6040	8m19.78s	5771	10m31.23s	5525	6.3s
V9	4377	4m58.545s	4389	20m18.42s	6004	8m19.78s	5794	10m31.23s	5509	6.3s
V10	4367	4m58.545s	4370	20m18.42s	5957	8m19.78s	6013	10m31.23s	5301	6.3s
V11	4399	4m58.545s	4346	20m18.42s	5870	8m19.78s	6045	10m31.23s	5470	6.3s
V12	4362	4m58.545s	4357	20m18.42s	6036	8m19.78s	5967	10m31.23s	5453	6.3s
V13	4407	4m58.545s	4416	20m18.42s	6169	8m19.78s	6078	10m31.23s	5447	6.3s
V14	4423	4m58.545s	4379	20m18.42s	5878	8m19.78s	5965	10m31.23s	5431	6.3s
V15	4395	4m58.545s	4346	20m18.42s	5974	8m19.78s	5676	10m31.23s	5598	6.3s
V16	4363	4m58.545s	4411	20m18.42s	5808	8m19.78s	5864	10m31.23s	5384	6.3s
V17	4363	4m58.545s	4499	20m18.42s	5867	8m19.78s	5896	10m31.23s	5471	6.3s
V18	4406	4m58.545s	4390	20m18.42s	6053	8m19.78s	5832	10m31.23s	5619	6.3s
V19	4329	4m58.545s	4371	20m18.42s	6045	8m19.78s	5963	10m31.23s	5246	6.3s
V20	4384	4m58.545s	4356	20m18.42s	5892	8m19.78s	5935	10m31.23s	5419	6.3s
V21	4374	4m58.545s	4381	20m18.42s	5882	8m19.78s	5825	10m31.23s	5371	6.3s
V22	4364	4m58.545s	4343	20m18.42s	5948	8m19.78s	5737	10m31.23s	5361	6.3s
V23	4299	4m58.545s	4322	20m18.42s	6083	8m19.78s	6081	10m31.23s	5428	6.3s
V24	4429	4m58.545s	4432	20m18.42s	5906	8m19.78s	5950	10m31.23s	5621	6.3s
V25	4425	4m58.545s	4381	20m18.42s	6022	8m19.78s	5882	10m31.23s	5357	6.3s
V26	4355	4m58.545s	4383	20m18.42s	6115	8m19.78s	5744	10m31.23s	5399	6.3s
V27	4417	4m58.545s	4332	20m18.42s	6022	8m19.78s	5952	10m31.23s	5318	6.3s
V28	4328	4m58.545s	4390	20m18.42s	5927	8m19.78s	5830	10m31.23s	5303	6.3s
V29	4376	4m58.545s	4349	20m18.42s	5884	8m19.78s	5828	10m31.23s	5515	6.3s
V30	4336	4m58.545s	4382	20m18.42s	6054	8m19.78s	5913	10m31.23s	5342	6.3s
V31	4389	4m58.545s	4408	20m18.42s	6028	8m19.78s	5681	10m31.23s	5631	6.3s
V32	4390	4m58.545s	4395	20m18.42s	5895	8m19.78s	5913	10m31.23s	5532	6.3s
V33	4432	4m58.545s	4443	20m18.42s	5978	8m19.78s	5915	10m31.23s	5441	6.3s
V34	4362	4m58.545s	4351	20m18.42s	6241	8m19.78s	5905	10m31.23s	5482	6.3s
V35	4407	4m58.545s	4343	20m18.42s	6104	8m19.78s	5903	10m31.23s	5401	6.3s
V36	4404	4m58.545s	4406	20m18.42s	5948	8m19.78s	5674	10m31.23s	5786	6.3s
V37	4323	4m58.545s	4382	20m18.42s	6086	8m19.78s	5871	10m31.23s	5283	6.3s
V38	4366	4m58.545s	4318	20m18.42s	5912	8m19.78s	5743	10m31.23s	5639	6.3s
V39	4431	4m58.545s	4403	20m18.42s	5863	8m19.78s	5766	10m31.23s	5481	6.3s
V40	4385	4m58.545s	4386	20m18.42s	5947	8m19.78s	5925	10m31.23s	5353	6.3s
V41	4443	4m58.545s	4413	20m18.42s	5942	8m19.78s	5983	10m31.23s	5563	6.3s
V42	4382	4m58.545s	4367	20m18.42s	5836	8m19.78s	5937	10m31.23s	5289	6.3s
V43	4338	4m58.545s	4343	20m18.42s	5836	8m19.78s	5742	10m31.23s	5588	6.3s
V44	4309	4m58.545s	4292	20m18.42s	5963	8m19.78s	6012	10m31.23s	5417	6.3s
V45	4354	4m58.545s	4355	20m18.42s	6155	8m19.78s	5845	10m31.23s	5608	6.3s
V46	4398	4m58.545s	4396	20m18.42s	5857	8m19.78s	5819	10m31.23s	5477	6.3s
V47	4337	4m58.545s	4370	20m18.42s	6081	8m19.78s	5749	10m31.23s	5555	6.3s
V48	4365	4m58.545s	4351	20m18.42s	6022	8m19.78s	5908	10m31.23s	5638	6.3s
V49	4422	4m58.545s	4391	20m18.42s	5931	8m19.78s	5744	10m31.23s	5489	6.3s
V50	4401	4m58.545s	4489	20m18.42s	6199	8m19.78s	5949	10m31.23s	5404	6.3s

Tabela 13 - Resultados para o G600

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G700	**4570**	8m46.2s	**4570**	38m39.3s	6724	16m1.55s	**6381**	20m17.5s	5896	10.19s
V1	4772	8m46.2s	4785	38m39.3s	6706	16m1.55s	6600	20m17.5s	6161	10.19s
V2	4848	8m46.2s	4835	38m39.3s	6865	16m1.55s	6736	20m17.5s	6308	10.19s
V3	4732	8m46.2s	4668	38m39.3s	6915	16m1.55s	6658	20m17.5s	6449	10.19s
V4	4755	8m46.2s	4785	38m39.3s	6738	16m1.55s	6437	20m17.5s	6141	10.19s
V5	4777	8m46.2s	4781	38m39.3s	6962	16m1.55s	6521	20m17.5s	6147	10.19s
V6	4788	8m46.2s	4867	38m39.3s	6900	16m1.55s	6567	20m17.5s	6123	10.19s
V7	4677	8m46.2s	4687	38m39.3s	7040	16m1.55s	6719	20m17.5s	6195	10.19s
V8	4865	8m46.2s	4869	38m39.3s	6633	16m1.55s	6530	20m17.5s	6103	10.19s
V9	4724	8m46.2s	4718	38m39.3s	6805	16m1.55s	6629	20m17.5s	6005	10.19s
V10	4740	8m46.2s	4751	38m39.3s	6809	16m1.55s	6689	20m17.5s	6106	10.19s
V11	4794	8m46.2s	4766	38m39.3s	6703	16m1.55s	6646	20m17.5s	6179	10.19s
V12	4691	8m46.2s	4720	38m39.3s	6817	16m1.55s	6436	20m17.5s	**5889**	10.19s
V13	4745	8m46.2s	4804	38m39.3s	6882	16m1.55s	6624	20m17.5s	6176	10.19s
V14	4743	8m46.2s	4721	38m39.3s	6794	16m1.55s	6528	20m17.5s	5978	10.19s
V15	4743	8m46.2s	4778	38m39.3s	6821	16m1.55s	6516	20m17.5s	6135	10.19s
V16	4717	8m46.2s	4714	38m39.3s	6773	16m1.55s	6693	20m17.5s	6015	10.19s
V17	4742	8m46.2s	4721	38m39.3s	6873	16m1.55s	6592	20m17.5s	6138	10.19s
V18	4776	8m46.2s	4816	38m39.3s	6683	16m1.55s	6716	20m17.5s	6326	10.19s
V19	4784	8m46.2s	4764	38m39.3s	6599	16m1.55s	6609	20m17.5s	6199	10.19s
V20	4748	8m46.2s	4801	38m39.3s	6787	16m1.55s	6497	20m17.5s	6060	10.19s
V21	4727	8m46.2s	4767	38m39.3s	6754	16m1.55s	6489	20m17.5s	6021	10.19s
V22	4736	8m46.2s	4821	38m39.3s	6772	16m1.55s	6769	20m17.5s	6042	10.19s
V23	4796	8m46.2s	4724	38m39.3s	6927	16m1.55s	6532	20m17.5s	6084	10.19s
V24	4750	8m46.2s	4741	38m39.3s	6893	16m1.55s	6703	20m17.5s	6023	10.19s
V25	4803	8m46.2s	4801	38m39.3s	6718	16m1.55s	6465	20m17.5s	6284	10.19s
V26	4850	8m46.2s	4849	38m39.3s	6757	16m1.55s	6691	20m17.5s	6044	10.19s
V27	4706	8m46.2s	4817	38m39.3s	6699	16m1.55s	6498	20m17.5s	6071	10.19s
V28	4760	8m46.2s	4779	38m39.3s	6936	16m1.55s	6687	20m17.5s	6090	10.19s
V29	4846	8m46.2s	4825	38m39.3s	7023	16m1.55s	6591	20m17.5s	6249	10.19s
V30	4810	8m46.2s	4791	38m39.3s	6700	16m1.55s	6712	20m17.5s	6067	10.19s
V31	4733	8m46.2s	4837	38m39.3s	6735	16m1.55s	6692	20m17.5s	6049	10.19s
V32	4692	8m46.2s	4646	38m39.3s	6860	16m1.55s	6619	20m17.5s	5952	10.19s
V33	4793	8m46.2s	4775	38m39.3s	6810	16m1.55s	6774	20m17.5s	6305	10.19s
V34	4703	8m46.2s	4705	38m39.3s	6662	16m1.55s	6473	20m17.5s	6222	10.19s
V35	4740	8m46.2s	4791	38m39.3s	6802	16m1.55s	6528	20m17.5s	6189	10.19s
V36	4809	8m46.2s	4794	38m39.3s	6812	16m1.55s	6766	20m17.5s	5905	10.19s
V37	4663	8m46.2s	4724	38m39.3s	**6606**	16m1.55s	6696	20m17.5s	6058	10.19s
V38	4861	8m46.2s	4851	38m39.3s	6834	16m1.55s	6648	20m17.5s	5909	10.19s
V39	4706	8m46.2s	4655	38m39.3s	6784	16m1.55s	6626	20m17.5s	6192	10.19s
V40	4769	8m46.2s	4739	38m39.3s	6817	16m1.55s	6656	20m17.5s	6147	10.19s
V41	4816	8m46.2s	4860	38m39.3s	6974	16m1.55s	6647	20m17.5s	6280	10.19s
V42	4725	8m46.2s	4715	38m39.3s	6947	16m1.55s	6588	20m17.5s	6183	10.19s
V43	4834	8m46.2s	4808	38m39.3s	6805	16m1.55s	6517	20m17.5s	6089	10.19s
V44	4691	8m46.2s	4785	38m39.3s	6837	16m1.55s	6734	20m17.5s	6238	10.19s
V45	4847	8m46.2s	4809	38m39.3s	6799	16m1.55s	6426	20m17.5s	6187	10.19s
V46	4711	8m46.2s	4736	38m39.3s	6761	16m1.55s	6555	20m17.5s	6162	10.19s
V47	4793	8m46.2s	4753	38m39.3s	6718	16m1.55s	6490	20m17.5s	6091	10.19s
V48	4750	8m46.2s	4791	38m39.3s	7004	16m1.55s	6563	20m17.5s	6258	10.19s
V49	4684	8m46.2s	4770	38m39.3s	6981	16m1.55s	6445	20m17.5s	6158	10.19s
V50	4828	8m46.2s	4798	38m39.3s	7006	16m1.55s	6725	20m17.5s	6048	10.19s

Tabela 14 - Resultados para o G700

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G800	**5288**	13m47.23s	**5302**	71m56.78s	7697	31m20.06s	7413	36m5.87s	**6571**	15.81s
V1	5422	13m47.23s	5436	71m56.78s	7562	31m20.06s	7551	36m5.87s	6982	15.81s
V2	5432	13m47.23s	5423	71m56.78s	7595	31m20.06s	7400	36m5.87s	6975	15.81s
V3	5429	13m47.23s	5473	71m56.78s	7722	31m20.06s	7483	36m5.87s	7175	15.81s
V4	5431	13m47.23s	5469	71m56.78s	7626	31m20.06s	7513	36m5.87s	6982	15.81s
V5	5437	13m47.23s	5404	71m56.78s	7665	31m20.06s	7528	36m5.87s	7016	15.81s
V6	5476	13m47.23s	5458	71m56.78s	7793	31m20.06s	7517	36m5.87s	7029	15.81s
V7	5484	13m47.23s	5420	71m56.78s	7753	31m20.06s	7481	36m5.87s	6970	15.81s
V8	5491	13m47.23s	5465	71m56.78s	7744	31m20.06s	7385	36m5.87s	7004	15.81s
V9	5444	13m47.23s	5356	71m56.78s	**7466**	31m20.06s	7575	36m5.87s	6852	15.81s
V10	5508	13m47.23s	5475	71m56.78s	7904	31m20.06s	7504	36m5.87s	7053	15.81s
V11	5357	13m47.23s	5396	71m56.78s	7592	31m20.06s	7365	36m5.87s	6772	15.81s
V12	5337	13m47.23s	5449	71m56.78s	7653	31m20.06s	7395	36m5.87s	7141	15.81s
V13	5419	13m47.23s	5469	71m56.78s	7749	31m20.06s	**7237**	36m5.87s	7040	15.81s
V14	5426	13m47.23s	5466	71m56.78s	7804	31m20.06s	7838	36m5.87s	7191	15.81s
V15	5442	13m47.23s	5460	71m56.78s	7675	31m20.06s	7374	36m5.87s	6919	15.81s
V16	5383	13m47.23s	5443	71m56.78s	7599	31m20.06s	7391	36m5.87s	7048	15.81s
V17	5444	13m47.23s	5498	71m56.78s	7771	31m20.06s	7465	36m5.87s	7080	15.81s
V18	5528	13m47.23s	5438	71m56.78s	7727	31m20.06s	7520	36m5.87s	6901	15.81s
V19	5490	13m47.23s	5490	71m56.78s	7784	31m20.06s	7652	36m5.87s	6903	15.81s
V20	5470	13m47.23s	5405	71m56.78s	7673	31m20.06s	7628	36m5.87s	6796	15.81s
V21	5397	13m47.23s	5447	71m56.78s	7853	31m20.06s	7467	36m5.87s	6937	15.81s
V22	5465	13m47.23s	5429	71m56.78s	7612	31m20.06s	7546	36m5.87s	7315	15.81s
V23	5521	13m47.23s	5552	71m56.78s	7475	31m20.06s	7437	36m5.87s	7016	15.81s
V24	5440	13m47.23s	5394	71m56.78s	7621	31m20.06s	7699	36m5.87s	6902	15.81s
V25	5455	13m47.23s	5388	71m56.78s	7780	31m20.06s	7471	36m5.87s	7099	15.81s
V26	5448	13m47.23s	5510	71m56.78s	7573	31m20.06s	7539	36m5.87s	7270	15.81s
V27	5443	13m47.23s	5411	71m56.78s	7914	31m20.06s	7717	36m5.87s	7114	15.81s
V28	5400	13m47.23s	5394	71m56.78s	7969	31m20.06s	7407	36m5.87s	7020	15.81s
V29	5389	13m47.23s	5447	71m56.78s	7930	31m20.06s	7471	36m5.87s	7078	15.81s
V30	5569	13m47.23s	5500	71m56.78s	7689	31m20.06s	7318	36m5.87s	6925	15.81s
V31	5452	13m47.23s	5424	71m56.78s	7978	31m20.06s	7480	36m5.87s	7073	15.81s
V32	5433	13m47.23s	5446	71m56.78s	7584	31m20.06s	7470	36m5.87s	6649	15.81s
V33	5454	13m47.23s	5513	71m56.78s	7594	31m20.06s	7544	36m5.87s	6836	15.81s
V34	5439	13m47.23s	5422	71m56.78s	7647	31m20.06s	7419	36m5.87s	7090	15.81s
V35	5459	13m47.23s	5414	71m56.78s	7731	31m20.06s	7729	36m5.87s	6937	15.81s
V36	5421	13m47.23s	5441	71m56.78s	7664	31m20.06s	7457	36m5.87s	7051	15.81s
V37	5340	13m47.23s	5413	71m56.78s	7660	31m20.06s	7335	36m5.87s	7005	15.81s
V38	5452	13m47.23s	5337	71m56.78s	7672	31m20.06s	7427	36m5.87s	7043	15.81s
V39	5436	13m47.23s	5455	71m56.78s	7685	31m20.06s	7324	36m5.87s	6837	15.81s
V40	5499	13m47.23s	5477	71m56.78s	7677	31m20.06s	7514	36m5.87s	7064	15.81s
V41	5458	13m47.23s	5509	71m56.78s	7895	31m20.06s	7493	36m5.87s	6997	15.81s
V42	5403	13m47.23s	5417	71m56.78s	7908	31m20.06s	7351	36m5.87s	7103	15.81s
V43	5511	13m47.23s	5431	71m56.78s	7655	31m20.06s	7540	36m5.87s	6859	15.81s
V44	5444	13m47.23s	5471	71m56.78s	7661	31m20.06s	7509	36m5.87s	6845	15.81s
V45	5387	13m47.23s	5383	71m56.78s	7797	31m20.06s	7660	36m5.87s	7067	15.81s
V46	5466	13m47.23s	5500	71m56.78s	7775	31m20.06s	7464	36m5.87s	6978	15.81s
V47	5506	13m47.23s	5523	71m56.78s	7856	31m20.06s	7535	36m5.87s	6930	15.81s
V48	5436	13m47.23s	5409	71m56.78s	7504	31m20.06s	7341	36m5.87s	6877	15.81s
V49	5497	13m47.23s	5541	71m56.78s	7633	31m20.06s	7714	36m5.87s	7077	15.81s
V50	5460	13m47.23s	5484	71m56.78s	7576	31m20.06s	7639	36m5.87s	6949	15.81s

Tabela 15 - Resultados para o G800

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G900	**5550**	26m14.228s	**5629**	118m55.92s	8231	70m44.56s	8105	73m29.14s	7388	28.77s
V1	5740	26m14.228s	5779	118m55.92s	8480	70m44.56s	8014	73m29.14s	7675	28.77s
V2	5743	26m14.228s	5775	118m55.92s	8253	70m44.56s	8102	73m29.14s	7699	28.77s
V3	5779	26m14.228s	5751	118m55.92s	8481	70m44.56s	8227	73m29.14s	7720	28.77s
V4	5775	26m14.228s	5751	118m55.92s	8273	70m44.56s	7926	73m29.14s	7769	28.77s
V5	5826	26m14.228s	5710	118m55.92s	8390	70m44.56s	8030	73m29.14s	7699	28.77s
V6	5749	26m14.228s	5740	118m55.92s	8513	70m44.56s	8347	73m29.14s	7557	28.77s
V7	5962	26m14.228s	5795	118m55.92s	8394	70m44.56s	8198	73m29.14s	7623	28.77s
V8	5779	26m14.228s	5747	118m55.92s	8474	70m44.56s	8285	73m29.14s	7710	28.77s
V9	5872	26m14.228s	5870	118m55.92s	8309	70m44.56s	8049	73m29.14s	7540	28.77s
V10	5762	26m14.228s	5734	118m55.92s	8654	70m44.56s	8228	73m29.14s	7656	28.77s
V11	5714	26m14.228s	5761	118m55.92s	8633	70m44.56s	8174	73m29.14s	7700	28.77s
V12	5759	26m14.228s	5804	118m55.92s	8531	70m44.56s	8102	73m29.14s	7495	28.77s
V13	5773	26m14.228s	5737	118m55.92s	8181	70m44.56s	8105	73m29.14s	7743	28.77s
V14	5782	26m14.228s	5814	118m55.92s	**8076**	70m44.56s	8173	73m29.14s	7692	28.77s
V15	5788	26m14.228s	5860	118m55.92s	8327	70m44.56s	8303	73m29.14s	7634	28.77s
V16	5839	26m14.228s	5794	118m55.92s	8602	70m44.56s	8175	73m29.14s	7756	28.77s
V17	5793	26m14.228s	5811	118m55.92s	8698	70m44.56s	8213	73m29.14s	7724	28.77s
V18	5841	26m14.228s	5839	118m55.92s	8407	70m44.56s	8350	73m29.14s	7814	28.77s
V19	5832	26m14.228s	5893	118m55.92s	8343	70m44.56s	8223	73m29.14s	7633	28.77s
V20	5659	26m14.228s	5697	118m55.92s	8360	70m44.56s	8080	73m29.14s	7532	28.77s
V21	5724	26m14.228s	5733	118m55.92s	8332	70m44.56s	8270	73m29.14s	7362	28.77s
V22	5818	26m14.228s	5803	118m55.92s	8163	70m44.56s	7898	73m29.14s	8129	28.77s
V23	5842	26m14.228s	5846	118m55.92s	8389	70m44.56s	8143	73m29.14s	7601	28.77s
V24	5784	26m14.228s	5764	118m55.92s	8509	70m44.56s	8164	73m29.14s	**7352**	28.77s
V25	5790	26m14.228s	5818	118m55.92s	8526	70m44.56s	8065	73m29.14s	7515	28.77s
V26	5794	26m14.228s	5739	118m55.92s	8508	70m44.56s	8010	73m29.14s	7521	28.77s
V27	5748	26m14.228s	5827	118m55.92s	8277	70m44.56s	8243	73m29.14s	7651	28.77s
V28	5820	26m14.228s	5803	118m55.92s	8346	70m44.56s	8193	73m29.14s	7760	28.77s
V29	5778	26m14.228s	5831	118m55.92s	8381	70m44.56s	8264	73m29.14s	7824	28.77s
V30	5777	26m14.228s	5847	118m55.92s	8367	70m44.56s	8281	73m29.14s	7558	28.77s
V31	5832	26m14.228s	5816	118m55.92s	8246	70m44.56s	8184	73m29.14s	7754	28.77s
V32	5782	26m14.228s	5805	118m55.92s	8381	70m44.56s	8062	73m29.14s	7698	28.77s
V33	5684	26m14.228s	5798	118m55.92s	8315	70m44.56s	8142	73m29.14s	7606	28.77s
V34	5806	26m14.228s	5850	118m55.92s	8398	70m44.56s	8431	73m29.14s	7687	28.77s
V35	5819	26m14.228s	5836	118m55.92s	8439	70m44.56s	8315	73m29.14s	7734	28.77s
V36	5752	26m14.228s	5792	118m55.92s	8440	70m44.56s	8103	73m29.14s	7806	28.77s
V37	5741	26m14.228s	5749	118m55.92s	8316	70m44.56s	8256	73m29.14s	7682	28.77s
V38	5810	26m14.228s	5765	118m55.92s	8594	70m44.56s	8051	73m29.14s	7782	28.77s
V39	5815	26m14.228s	5737	118m55.92s	8417	70m44.56s	8185	73m29.14s	7550	28.77s
V40	5774	26m14.228s	5781	118m55.92s	8252	70m44.56s	8213	73m29.14s	7770	28.77s
V41	5774	26m14.228s	5782	118m55.92s	8489	70m44.56s	8294	73m29.14s	7558	28.77s
V42	5749	26m14.228s	5791	118m55.92s	8407	70m44.56s	8295	73m29.14s	7713	28.77s
V43	5743	26m14.228s	5797	118m55.92s	8664	70m44.56s	7997	73m29.14s	7811	28.77s
V44	5790	26m14.228s	5862	118m55.92s	8360	70m44.56s	8146	73m29.14s	7692	28.77s
V45	5682	26m14.228s	5728	118m55.92s	8202	70m44.56s	8368	73m29.14s	7638	28.77s
V46	5748	26m14.228s	5700	118m55.92s	8381	70m44.56s	8000	73m29.14s	7563	28.77s
V47	5795	26m14.228s	5760	118m55.92s	8525	70m44.56s	8309	73m29.14s	7747	28.77s
V48	5783	26m14.228s	5839	118m55.92s	8467	70m44.56s	8172	73m29.14s	7706	28.77s
V49	5807	26m14.228s	5786	118m55.92s	8270	70m44.56s	8231	73m29.14s	7546	28.77s
V50	5835	26m14.228s	5838	118m55.92s	8256	70m44.56s	**7861**	73m29.14s	7702	28.77s

Tabela 16 - Resultados para o G900

Grafo	H1M		H1		HIMP		HIMB		HVMP	
	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
G1000	**6519**	41m3.92s	**6555**	190m53.37s	9228	109m43.54s	9299	119m5.35s	8612	41.5s
V1	6695	41m3.92s	6739	190m53.37s	9628	109m43.54s	9426	119m5.35s	8728	41.5s
V2	6784	41m3.92s	6713	190m53.37s	9620	109m43.54s	9201	119m5.35s	8684	41.5s
V3	6746	41m3.92s	6698	190m53.37s	9648	109m43.54s	9533	119m5.35s	8865	41.5s
V4	6692	41m3.92s	6692	190m53.37s	9617	109m43.54s	9401	119m5.35s	8796	41.5s
V5	6795	41m3.92s	6763	190m53.37s	9374	109m43.54s	9209	119m5.35s	8824	41.5s
V6	6739	41m3.92s	6725	190m53.37s	9679	109m43.54s	9417	119m5.35s	8681	41.5s
V7	6718	41m3.92s	6657	190m53.37s	9549	109m43.54s	9287	119m5.35s	8968	41.5s
V8	6780	41m3.92s	6745	190m53.37s	9573	109m43.54s	9393	119m5.35s	8808	41.5s
V9	6748	41m3.92s	6723	190m53.37s	9630	109m43.54s	9547	119m5.35s	8846	41.5s
V10	6718	41m3.92s	6753	190m53.37s	9704	109m43.54s	9275	119m5.35s	8920	41.5s
V11	6675	41m3.92s	6645	190m53.37s	9669	109m43.54s	9173	119m5.35s	8622	41.5s
V12	6783	41m3.92s	6807	190m53.37s	9733	109m43.54s	9416	119m5.35s	8725	41.5s
V13	6751	41m3.92s	6738	190m53.37s	9699	109m43.54s	9188	119m5.35s	8905	41.5s
V14	6805	41m3.92s	6768	190m53.37s	9399	109m43.54s	9261	119m5.35s	8819	41.5s
V15	6718	41m3.92s	6725	190m53.37s	9876	109m43.54s	9316	119m5.35s	**8477**	41.5s
V16	6748	41m3.92s	6737	190m53.37s	9712	109m43.54s	9087	119m5.35s	8969	41.5s
V17	6685	41m3.92s	6685	190m53.37s	9563	109m43.54s	9119	119m5.35s	8910	41.5s
V18	6816	41m3.92s	6696	190m53.37s	**9136**	109m43.54s	**8944**	119m5.35s	8939	41.5s
V19	6700	41m3.92s	6716	190m53.37s	9561	109m43.54s	9371	119m5.35s	8723	41.5s
V20	6753	41m3.92s	6746	190m53.37s	9321	109m43.54s	9218	119m5.35s	8660	41.5s
V21	6787	41m3.92s	6709	190m53.37s	9509	109m43.54s	9450	119m5.35s	8759	41.5s
V22	6731	41m3.92s	6832	190m53.37s	9528	109m43.54s	9121	119m5.35s	8584	41.5s
V23	6693	41m3.92s	6756	190m53.37s	9685	109m43.54s	9111	119m5.35s	8774	41.5s
V24	6829	41m3.92s	6802	190m53.37s	9467	109m43.54s	9207	119m5.35s	8768	41.5s
V25	6711	41m3.92s	6775	190m53.37s	9589	109m43.54s	9449	119m5.35s	8957	41.5s
V26	6688	41m3.92s	6655	190m53.37s	9644	109m43.54s	9197	119m5.35s	8791	41.5s
V27	6753	41m3.92s	6700	190m53.37s	9360	109m43.54s	9308	119m5.35s	8833	41.5s
V28	6753	41m3.92s	6818	190m53.37s	9485	109m43.54s	9436	119m5.35s	8711	41.5s
V29	6666	41m3.92s	6626	190m53.37s	9461	109m43.54s	9363	119m5.35s	8947	41.5s
V30	6613	41m3.92s	6602	190m53.37s	9588	109m43.54s	9251	119m5.35s	8931	41.5s
V31	6756	41m3.92s	6798	190m53.37s	9808	109m43.54s	9334	119m5.35s	8911	41.5s
V32	6716	41m3.92s	6747	190m53.37s	9436	109m43.54s	9353	119m5.35s	8914	41.5s
V33	6656	41m3.92s	6691	190m53.37s	9645	109m43.54s	9074	119m5.35s	8924	41.5s
V34	6767	41m3.92s	6749	190m53.37s	9714	109m43.54s	9331	119m5.35s	8650	41.5s
V35	6692	41m3.92s	6650	190m53.37s	9551	109m43.54s	9317	119m5.35s	8826	41.5s
V36	6839	41m3.92s	6813	190m53.37s	9495	109m43.54s	9212	119m5.35s	8915	41.5s
V37	6688	41m3.92s	6662	190m53.37s	9391	109m43.54s	9077	119m5.35s	8902	41.5s
V38	6811	41m3.92s	6766	190m53.37s	9548	109m43.54s	9404	119m5.35s	8813	41.5s
V39	6687	41m3.92s	6709	190m53.37s	9695	109m43.54s	9404	119m5.35s	9086	41.5s
V40	6683	41m3.92s	6685	190m53.37s	9593	109m43.54s	9180	119m5.35s	8756	41.5s
V41	6719	41m3.92s	6768	190m53.37s	9385	109m43.54s	9174	119m5.35s	8631	41.5s
V42	6727	41m3.92s	6746	190m53.37s	9748	109m43.54s	9280	119m5.35s	8812	41.5s
V43	6726	41m3.92s	6719	190m53.37s	9620	109m43.54s	9275	119m5.35s	8803	41.5s
V44	6708	41m3.92s	6660	190m53.37s	9768	109m43.54s	9350	119m5.35s	8736	41.5s
V45	6730	41m3.92s	6713	190m53.37s	9512	109m43.54s	9448	119m5.35s	8815	41.5s
V46	6657	41m3.92s	6664	190m53.37s	9588	109m43.54s	9071	119m5.35s	8933	41.5s
V47	6755	41m3.92s	6811	190m53.37s	9703	109m43.54s	9215	119m5.35s	8768	41.5s
V48	6683	41m3.92s	6673	190m53.37s	9632	109m43.54s	9220	119m5.35s	8913	41.5s
V49	6835	41m3.92s	6779	190m53.37s	9507	109m43.54s	9235	119m5.35s	8573	41.5s
V50	6759	41m3.92s	6801	190m53.37s	9542	109m43.54s	9338	119m5.35s	8816	41.5s

Tabela 17-Resultados para o G1000

8. Conclusões

Neste trabalho, pode-se perceber o quão interessante é o PCVG, tanto pelo aspecto prático, pois diversos problemas reais se reportam ao PCVG, quanto pelo aspecto teórico, por fornecer informações importantes sobre o comportamento de heurísticas e metaheurísticas para o PCV e suas extensões.

Como foi dito anteriormente, o PCVG é um problema NP-árduo. Portanto, embora fácil de se definir, é difícil de se resolver. De onde surge a idéia de aplicar heurísticas e metaheurísticas mais modernas na solução do PCVG.

As duas metaheurísticas consideradas, GRASP e VNS, trabalham em duas etapas, inicialmente utilizam um método de construção para encontrar uma solução inicial, e a partir disto, utilizando cada uma delas, suas características próprias, procuram melhorar a solução inicial. Com as idéias propostas por Chisman (1975) associadas aos métodos de construção (GENI) e busca local (US) definidos por Gendreau (1992), foi possível elaborar diversos algoritmos que aplicados ao PCVG obtiveram bons resultados, confirmando a grande aplicabilidade destes métodos.

Um componente comum entre os algoritmos implementados foi o uso de GENI como o método de construção da elaboração da solução inicial. Por outro lado, as várias versões desses algoritmos se diferenciam, basicamente, pelo método de busca escolhido na fase de busca local. Nesse sentido, a contribuição desse trabalho foi mostrar que através de modificações na estrutura do método de busca US é possível obter novos algoritmos e, com eles, obter boas soluções para o PCVG, tanto em relação ao tempo computacional, quanto em relação à qualidade da solução obtida.

É importante destacar que foram identificadas diversas estruturas possíveis de busca que poderiam ser definidas a partir do método US associado ao VNS. Portanto, entre as

estruturas escolhidas, foram apresentadas somente as que obtiveram um resultado médio de melhor qualidade. Entretanto, futuramente, dando continuidade a esse estudo, novos algoritmos podem ser elaborados através de novas modificações.

Uma das idéias propostas neste trabalho foi a de resolver o PCVG através de suas variações. O esforço realizado na implementação das heurísticas de construção foi bastante válido na avaliação desta idéia. Com os resultados obtidos, pode-se constatar que, na maioria dos casos, as soluções obtidas com as variações, com pelo menos uma dessas heurísticas, foram melhores que as soluções obtidas com a instância original. O que motivou a elaboração das versões do GRASP e VNS que consideram essas variações. O problema encontrado nessas versões foi o tempo de execução exigido, que naturalmente, cresce proporcionalmente ao número de variações consideradas. Portanto, em casos onde o tempo computacional não é o fator mais importante de avaliação, esses algoritmos foram mais eficientes.

Pelos resultados obtidos, podemos concluir que uma fórmula para que uma metaheurística obtenha uma busca eficiente é perturbando os dados de entrada ao invés de modificar seus parâmetros.

Neste contexto, um trabalho futuro seria elaborar versões desses algoritmos de maneira que as variações fossem resolvidas distribuídas em diversos processadores. O que não é uma tarefa muito difícil devido à própria estrutura dos algoritmos e pelo fato de que o paralelismo é um recurso bastante usado em métodos desenvolvidos com o GRASP.

Finalmente, outro aspecto interessante intrínseco aos métodos desenvolvidos para o PCVG é a maneira como o problema definido entre os vértices e como o problema definido entre os grupamentos do PCVG são solucionados. Embora em alguns métodos descritos, esses problemas foram resolvidos separadamente, em todos eles a solução do

problema definido entre os grupamentos era obtida à medida que o problema entre os vértices era solucionado. Deste modo, deduz-se que esta é a melhor abordagem para os métodos onde esses problemas são resolvidos separadamente.

Esta característica fica ainda mais clara quando se observa os resultados obtidos com a primeira versão do GRASP que foi apresentada. Nesta versão, o PCVG foi resolvido determinando, primeiro, a seqüência de visitas entre os grupamentos e, a partir desse resultado, a seqüência de visita entre os vértices. Como descrito no capítulo anterior, essa versão do GRASP não obteve boas soluções quando comparada aos outros algoritmos. Deste modo, outra sugestão para novos trabalhos seria definir um GRASP onde esses problemas fossem resolvidos de maneira que a solução do problema entre os vértices determinasse a solução do problema entre os grupamentos.

Referências Bibliográficas

ANILY, S., BRAMEL, J., 1999a, "Approximation Algorithms for the Capacitated Traveling Salesman Problem with Pickups and Deliveries", *Naval Research Logistics*, v. 46, pp 654-670.

ANILY S, BRAMEL. J., HERTZ A., 1999b, "A 5/3–Approximation Algorithm for the Clustered Traveling Salesman tour and path problems", *Operations Research Letters*, v. 24, pp 29-35.

ARKIN, E. M., HASSIN, R., KLEIN, L., 1997, "Restricted Delivery Problem on a Network". *Network*, 29, pp 205–216.

BARD, J.F., FEO, T.A, 1989, "Operations sequencing in discrete parts manufacturing", *Management Science*, v 35, pp 249-255.

CASCO D. O., GOLDEN, B. L., WASIL., E. A., 1988, "Vehicle routing with Backhauls", In: Golden B. L., Assad, A. A. (eds), *Vehicle Routing: Methods and Studies*), pp127-147, Amsterdam, North-Holland.

CHISMAN, J. A., 1975, "The clustered salesman problem", *Computer & Operations Research*, v 2, pp 115–119.

CHRISTOFIDES, N., 1976, "Worst-case analysis of a new heuristic for the traveling salesman problem", In: Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

DEIF, I., BODIN, L.D., 1984, Extension of the Clarke and Wright algorithm for solving the vehicle routing with backhauling. *Proceedings of the Babson Conference on*

Software Uses in Transportation and Logistics Management, Kidder, A. E.(eds), pp 75-96, Babson Park, Mass.

EASTMAN, W. L., 1958, *Linear Programming with Pattern Constraints*, Ph. D. Dissertation, University of Harvard.

EISELT, H. A., GENDREAU, M., LAPORTE, G., 1995, "Arc routing problems, part II: The rural postman problem", *Operations Research* , v 43, pp 399-414.

FEO, T. A., RESENDE, M. G. C, 1995, "Greedy Randomized adaptive search procedures", *Journal of Global Optimization*, v 6, pp 109-133.

FEO, T. A., RESENDE, M. G. C., SMITH, S. H, 1994, "A greedy randomized adaptive search for maximum independent set", *Operation Research*, v 42, pp 860-878.

FEO, T. A., VENKATRAMAN, K., BARD, J.F, 1991, "A GRASP for a difficult single machine scheduling problem", *Computer & Operations Research*, v.18, pp 635-643.

FREDERICKSON, G. N., 1979, "Approximation algorithms for some postman problems", *J. Assoc. Comput. Mach. JACM*, v 26, n 3, 538-554.

FREDERICKSON, G. N., HECHT, M. S., KIM, C. E., 1978, "Approximation algorithms for some routing problems", *SIAM Journal on Computing*, v 7, n 2, pp 178-193.

FREDMAN, M.L., JOHNSON, D.S, GEOCH, M. C., OSTHEIMER, G., 1995, "Data structures for traveling salesmen", Journal of Algorithms, v. 18, pp 432-479.

GELINAS, S., DESROCHERS, M., DESROSIERS, J., SOLOMON, M. M., 1992, *Vehicle routing with backhauling*, GERAD #G-92-13, École des Hautes Études Commerciales de Montreal.

GENDREAU, M., HERTZ, A., LAPORT, G.,1997, "An Approximation Algorithms for the Traveling Salesman Problem with Backhauls". *Operations Research*, v. 45, pp 639–641.

GENDREAU, M., HERTZ, A., LAPORT, G., 1996a, "The Traveling Salesman Problem with Backhauls". *Computer & Operation Research*, v 23, pp 501-508.

GENDREAU, M., HERTZ, A., LAPORT, G., 1992, "New insertion and postoptimization procedures for the traveling salesman problem", *Operations Research*, v 40, pp 1086-1094.

GENDREAU, M., LAPORTE, G., POTVIN, J.Y., 1996b, "Heuristics for the Clustered Traveling Salesman Problem". *Combinatorial Optimization,* v 1, pp 41–56.

GOETSCHALCKX, M., JACOBS-BLECHA, C., 1989, "The vehicle routing problem with backhauls", *European Journal of Operations Research*, v 42, pp 39–51.

GOLDEN, B. L., BAKER, E., ALFARO, J., SCHAFFER, J., 1985, "The vehicle routing problem with backhauling: two approaches", *Proceedings of the Twenty-First Annual Meeting of the* S. *S. TIMS,* Hammesfahr, R. D. (eds), pp 90-92, Myrtle Beach, S. C.

GUTTMAN-BECK, N., R.HASSIN, S. KHULLER, B. RAGHAVACHARI, 2000, "Approximation Algorithms with Bounded Performance Guarantees for the Clustered Traveling Salesman Problem", Algorithmica, v 28, pp 422-437.

HELD. M., 1971, "The Traveling-Salesman problem and minimum spanning trees: part II", *Mathematical Programming (Netherlands)*, v 1, n 1, pp 6-25.

HOOGEVEEN, J. A., 1991, "Analysis of Christofides heuristic: Some paths are more difficult than cycles," *Operations Research Letters*, v 10, n 5, pp 291-295.

JANSEN, K., 1992, "An approximation algorithm for the genenral routing problem," *Information Processing Letters*, v 41, pp 333-339.

JOHNSON, D. S., PAPADIMITRIOU, C. H., 1985, "Performance guarantees for heuristics", In: Lawer, E. L., Lentra, J. K., Rinnooy Kan, A. H. G., Shmoy, D. B. (eds), *The Traveling Salesman Problem: A guided tour of combinatorial optimization*, pp 145-180, Wiley & Sons.

JONGENS K., VOLGENATE T., 1985, "The Symmetric Clustered Traveling Salesman problem", *European Journal of Operations Research*, v 19, pp 68 – 75.

KINDERVATER, G.A.P., LENSTRA, J. K., SHMOYS, D.B, 1989, "The parallel complexity of TSP heuristics", *Journal of Algorithms*, v. 10, n. 2, pp 249-270.

LAGUNA, M., FEO, T.A., ELROD, H.C, 1994, "A greedy randomized adaptive search procedure for the two partition problem", *Operations Research*, v.42, pp 677-687.

LIN. S., KERNIGHAN, B. W., 1973, "An effective heuristic algorithm for the traveling salesman problem", *Operations Research*, v 21, n 2, pp 498–515.

LITTLE, J.D.C., MURTY, K. G., SWEENEY, D.W, KAREL, C., 1963, "An Algorithm for the Traveling Salesman Problem", *Operations Research*, v 11, n 2, pp 972-989..

LOKIN, F. C. J., 1978, "Procedures for traveling salesman problems with additional constraints", *European Journal of Operations Research*, v 3, pp 135-141.

MARTINS, S. L., PARDALOS, P.M, RESENDE, M.G.C., RIBEIRO, C.C., 1999b, *A parallel GRASP for the Steiner problem in graphs using a hybrid local search strategy*, In: Report AT&T Labs Research.

MARTINS, S. L., PARDALOS, P.M., RESENDE M.G.C., RIBEIRO, C.C., 1999a, "Greedy randomized adaptive search procedures for the Steiner problem in graphs", In: *Randomization methods in algorithmic design*, Pardalos, P.M., Rajasekaran, S., Rolin J. (eds), v. 43, *DIMACS Series on Discrete Mathematics an Theoretical Computer Science*, Sociedade Americana de Matemática, pp 133-145.

MLANDENOVIC, N., HANSEN, P., 1997, "Variable Neighborhood Search", *Computer Operational Research*, v. 24, n. 11, pp 1097–1100.

PARDALOS, P.M., PITSOULIS, L.S., RESENDE, M.G.C., 1996, A parallel GRASP for MAX_SAT problems. *Lecture Notes in Computer Science*, 1184: 575-585.

PARDALOS, P.M., RAPPE, J., RESENDE, M. G. C, 1995, "An exact parallel algorithm for the maximum clique problem", In: R., De Leone *et. Al..*, *High performance algorithms and software in nonlinear optimization*, pp 279-300, Kluwer Academic Publishers.

POTVIN, J. Y, GUERTIN, F., 1995, *A Genetic Algorithm for the Clustered Traveling Salesman Problem with a Priori Order on the Clustered*, Report CRT-95-05, Centre de recherche sur les transports, Universidade de Montreal.

POTVIN, J. Y, GUERTIN, F., 1996, "The Clustered Traveling Salesman Problem: A Genetic Approach ", In: Osman, I. H., Kelly J. P. (eds), *Metaheuristics: Theory and Applications*, chapter 37, pp 619-632, Norwell, MA, USA, Kluwer Academic Publishers.

RABELO, P.G., OCHI, L.S., 1997, "A new genetic metaheuristics fo Clustered Travelin Salesman Problem", *Proceedings of the MIC*, pp 59 – 64.

RESENDE, M. G. C., 1998, *Greedy Randomized adaptive search procedures (GRASP)*, Report NJ 07932, AT&T Labs Research, Florham Park, New Jersey, USA.

RESENDE, M. G. C., RIBEIRO, C. C., 2001, "Graph Planarization", *Encyclopedia of Optimization*, v 2, pp 368 – 373.

RESENDE, M. G. C., RIBEIRO, C.C., 1997, "A GRASP for graph planarization", *Networks*, v. 29, pp 173-189.

WHITLEY, D., STARKWEATHER, T., FUQUAY, D., 1989, "Scheduling problems and traveling salemen: the genetic edge recombination operator", *Proceedings of the Third International Conference on Genetic Algorithms*, Schaffer, J. D. (eds)., Morgan Kaufmann Publishers, San Mateo, CA, 133-140.