NOVAS PROPOSTAS PARA O PROBLEMA DE RECOBRIMENTO DE ROTAS

Luciana Roque Brito

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 DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Susana Scheimberg de Makler, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

Prof. Maria Helena Cautiero Horta Jardim, D.Sc.

Prof. Nelson Maculan Filho, D.Sc.

Prof. Henrique Pacca Loureiro Luna, D.Sc.

Prof. Lúcia Maria de Assumpção Drummond, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2005

BRITO, LUCIANA ROQUE

Novas Propostas para o Problema de Recobrimento de Rotas [Rio de Janeiro] 2005

VIII, 113 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2005)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Formulação Matemática e Relaxação Lagrangeana
2. Regras de Redução
3. Metaheurísticas GRASP e VNS

I. COPPE/UFRJ II. Titulo (Série)

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

NOVAS PROPOSTAS PARA O PROBLEMA DE RECOBRIMENTO DE ROTAS

Luciana Roque Brito

Junho/2005

Orientadores: Susana Scheimberg de Makler
Luiz Satoru Ochi

Programa: Engenharia de Sistemas e Computação

O Problema de Recobrimento de Rotas (PRR) é uma generalização do Problema do Caixeiro Viajante com várias aplicações reais. Seja $G = (N, E)$ um grafo não direcionado, sendo N o conjunto de vértices formado pela união de dois conjuntos disjuntos V e W, e $E = \{(v_i, v_j) \mid v_i, v_j \in N\}$ o conjunto de arestas. O conjunto V está associado aos vértices que podem ou não fazer parte da rota, ou seja, podem ou não ser visitados. T é o subconjunto de vértices de V que obrigatoriamente devem fazer parte da rota. W é o conjunto de vértices que devem ser cobertos pelos vértices do conjunto V que fazem parte da rota. A partir destas considerações, temos que o PRR consiste em determinar uma rota de comprimento mínimo ou um ciclo Hamiltoniano sobre um subconjunto de V, tal que todos os vértices do conjunto T ($T \subset V$) estejam na rota e todo vértice de W seja coberto por esta rota. Nesta tese, propomos e analisamos uma nova formulação matemática para o PRR, uma nova regra de redução e seis algoritmos heurísticos. No final deste trabalho, apresentamos os resultados computacionais para um conjunto de problemas teste.

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

NEW PROPOSALS FOR THE COVERING TOUR PROBLEM

Luciana Roque Brito

June/2005

Advisors: Susana Scheimberg de Makler
Luiz Satoru Ochi

Department: Systems Engineering and Computer Science

The Covering Tour Problem (CTP) is a generalization of the Traveling Salesman Problem (TSP) which has several actual applications. Let $G = (N, E)$ be an undirected graph, where N is the set of vertices resulting from the union of two disjointed sets V and W, and $E = \{(v_i, v_j) \mid v_i, v_j \in N\}$ is the set of edges. The set V is associated to the vertices that may or not belong to the tour, that is, the vertices that may or not be visited. T is the subset of vertices of V that must obligatorily be in the tour. W is the set of vertices that must be covered by the vertices of V which belong to the tour. From these considerations, we state the CTP as consisting in to determine a minimum length tour, or a Hamiltonian cycle, over a subset of V in such a way that all the vertices of the set T ($T \subset V$) must be into the tour and every vertex of W must be covered by the tour. In this Thesis, we propose and analyze a new mathematical formulation for the CTP, a new reduction rule and six heuristic algorithms. In the end of this work, we present the computational results for a set of test problems.

AGRADECIMENTOS

Ao Deus consolador e das vitórias, nosso “ajudador” em todos os momentos de adversidade.

“Fiz o melhor que pude na corrida, cheguei até o fim, conservei a fé”. (2Tm 4.7)

Ao meu amado José André, por não ter medido esforço algum para me ajudar nesta caminhada e em todos os outros momentos. Sem ele provavelmente não teria conseguido terminar esta etapa.

À minha mãe Selma e meu pai Milton, pela vida, pelo amor e por todos os ensinamentos.

À minha sobrinha Danielle, que mesmo com pouca idade soube entender minha ausência em vários momentos de sua vida.

À minha irmã, Márcia e ao meu cunhado, Daniel.

A todos os meus familiares e amigos.

À amiga Rosângela, pela ativa participação e ajuda desde o mestrado.

À amiga Fátima, pela sua eterna e inexorável presteza.

À professora Susana Scheimberg de Makler por ter aceitado ser a orientadora num momento “turbulento” pelo qual passamos.

Ao professor Luiz Satoru Ochi, por sua amizade e pela transmissão de seus ensinamentos e conhecimentos e, principalmente, pela força que nos deu em todos esses anos de trabalho.

A todos os funcionários e colegas do Programa de Engenharia de Sistemas e Computação.

À CAPES, pelo financiamento de minha pesquisa.

À Belinha e Mel.

Ao meu amado Marido, Companheiro e Amigo José André
e
a nossa querida e tão esperada filhinha Amanda

Índice

1 - Introdução

O Problema de Recobrimento de Rotas (PRR) é um problema de otimização combinatória de difícil solução e que está associado a um vasto conjunto de aplicações reais tais como: Planejamento de Serviços Médicos em áreas rurais; Transporte em Dois Níveis; Escala de Aeronaves e Localização de Postos de Coleta dos Correios.

Dado um grafo não direcionado $G = (N, E)$, sendo N o conjunto de vértices formado pela união de dois conjuntos V e W disjuntos, ou seja, $N = V \cup W$, $V \cap W = \emptyset$ e $E = \{(v_i, v_j) \mid v_i, v_j \in N\}$ o conjunto de arestas. O Problema de Recobrimento de Rotas (PRR) consiste em determinar uma rota de comprimento mínimo ou um ciclo Hamiltoniano (Swarficter[2]) sobre um subconjunto de V, tal que todos os vértices de T estejam na rota ($T \subset V$) e todos os vértices de W sejam cobertos por algum vértice desta rota.

Um vértice $w_l \in W$ é dito coberto se existir pelo menos um vértice da rota (vértice do conjunto T ou do conjunto $V \backslash T$) a uma distância de no máximo d ($d > 0$) deste vértice w_l , sendo d a distância de cobertura. Um vértice $v_j \in V \backslash T$ só estará na rota se ele for necessário para cobrir algum vértice w_l do conjunto W .

De acordo com a literatura, poucos trabalhos têm sido desenvolvidos para a resolução do PRR. O Problema de Recobrimento de Rotas foi primeiramente abordado por Current em 1981 [5] em sua tese de doutorado.

Em seu trabalho de 1995, Gendreau[1] desenvolveu uma formulação matemática para o PRR e uma heurística (*GENIUS* e PRIMAL1). Maniezzo[8] propos uma solução para o PRR através da aplicação de três algoritmos baseados na metaheurística *Scatter Search* (Glover[9]), desenvolveu regras de redução para PRR e uma formulação matemática.

Também encontramos na literatura, alguns trabalhos desenvolvidos para resolver problemas similares ao PRR tais como: *The Covering Salesman Problem, The Shortest Covering Path Problem,* etc.

Na tentativa de fornecer novas alternativas para a resolução do PRR, implementamos uma nova formulação matemática que utiliza variáveis de fluxo que evitam ciclos desconexos com a origem e um conjunto de algoritmos baseados nas metaheurísticas GRASP e VNS.

Além destes algoritmos e da nova formulação matemática, propomos uma nova regra de redução. Observando que aplicação desta nova regra não implica na perda de qualquer informação que leve a uma solução ótima para o PRR.

O desenvolvimento de nosso trabalho de tese está dividido em seis partes:

Na primeira parte (capítulo 2) temos uma descrição concisa do Problema de Recobrimento de Rotas e um panorama geral dos trabalhos que foram desenvolvidos para a resolução do PRR. A finalidade desta seção é a de fornecer o embasamento mínimo para o entendimento deste problema.

Dentre os trabalhos desenvolvidos para o PRR, apresentamos a formulação proposta por Gendreau *et al.*[6], a formulação proposta por Maniezzo *et al.*[8] e as quatro regras de redução propostas na literatura.

Na segunda parte (capítulo 3) temos a apresentação das seguintes propostas alternativas para a resolução do PRR:

- Uma nova formulação matemática que difere das existentes na literatura, por fazer uso de variáveis de fluxo para evitar ciclos desconexos da origem.

- A utilização da Relaxação Lagrangeana (Geoffrion[26]) considerando a nova formulação matemática proposta para o PRR, para a obtenção de limites inferiores melhores do que aqueles obtidos mediante a aplicação da relaxação linear (Bazaraa[27]).

- A implementação de uma Heurística Lagrangeana considerando as particularidades do PRR. Com a utilização desta heurística, "ajustamos " a solução obtida a partir da relaxação lagrangeana, de forma a obter uma solução viável que constituirá um limite superior para o PRR.

Na terceira parte (capítulo 4) temos a apresentação de uma nova regra de redução para o PRR. Esta regra é utilizada conjuntamente com as regras 3 e 4 da literatura para produzir os grafos reduzidos que serão utilizados como grafo de entrada pela nova formulação matemática e pelos algoritmos baseados no GRASP [34] e VNS [32,33].

Nesta regra, transformamos um vértice v_i do conjunto $V \backslash T$ em um vértice obrigatório t_k do conjunto T, se este vértice v_i for o único na cobertura de um vértice w_l do conjunto W.

Na quarta parte (capítulo 5), concluindo o conjunto de metodologias que foram implementadas para a resolução do PRR, temos a apresentação de seis algoritmos baseados no GRASP e VNS .

O objetivo destes algoritmos é o de possibilitar a obtenção de soluções viáveis, para diversas instâncias do PRR. Ademais, é possível utilizar o valor associado a estas soluções como um limite superior para a nova formulação matemática proposta para o PRR.

Considerando que o GRASP utiliza um procedimento na fase de construção e um procedimento na fase de busca local, implementamos e combinamos os seguintes algoritmos: Vizinho Mais Próximo, VNS, Troca Simultânea, Filtro e Reconexão por Caminhos (Path-Relinking). A combinação destes algoritmos proporcionou a obtenção de seis algoritmos GRASP denominados: GR1, GR2, GR1F, GR2F, GR1FPR-V0 e GR1FPR-V1.

Na quinta parte (capítulo 6) temos a descrição do programa implementado para gerar os grafos de entrada associados ao PRR (problemas teste), do programa que aplica as regras de redução apresentadas neste trabalho e de um conjunto de tabelas e gráficos que contemplam as seguintes informações:

- Impacto das regras de redução: Número de vértices reduzidos em média a partir da aplicação de dois conjuntos de regras de redução, utilizando 270 grafos com número de vértices variando de 10 a 500. O primeiro conjunto agregando as regras 3 e 4 (regras da literatura) e o segundo conjunto composto pelas regras nova, 3 e 4.

- Resultados da nova formulação matemática considerando um conjunto de 70 problemas teste. Estes resultados estão associados a solução ótima da formulação, a melhor solução viável associada a um dos seis algoritmos e a solução da relaxação linear.

- Resultados comparativos entre a nova formulação proposta neste trabalho e formulação de Maniezzo. Tais resultados dizem respeito ao número de restrições, número de variáveis binárias e tempo de processamento de cada uma das formulações, considerando um conjunto de 20 problemas teste.

- Resultados da Relaxação Lagrangeana e da Heurística Lagrangeana. Temos a apresentação das soluções viáveis geradas através da heurística e os limites inferiores obtidos a partir da relaxação.

Tais resultados estão associados às diferenças, em percentual, entre o valor ótimo da nova formulação e os valores da Relaxação Lagrangeana, da Relaxação Linear e da Heurística Lagrangeana.

- Análise das soluções viáveis obtidas a partir dos seis algoritmos propostos, considerando o conjunto de 70 problemas testes. Além das soluções referentes a cada algoritmo, podemos saber o valor da diferença entre a solução viável e a solução ótima, para os exemplos em que foi possível se obter o valor ótimo. Quando não foi possível obter a solução ótima, calculamos a diferença entre o valor da melhor solução (de um dos algoritmos) e a solução viável de cada algoritmo.

Para tal análise, consideramos o grafo original e os grafos reduzidos pelos conjuntos 1 e 2.

- Por último, avaliamos o desempenho dos seis algoritmos GRASP através da análise probabilística de dois problemas teste. Esta análise registra os tempos de processamento (t_i) que cada algoritmo utiliza, para atingir a um valor que seja menor ou igual ao alvo previamente estabelecido.

Na sexta parte (capítulo 7) temos as conclusões sobre o trabalho e as propostas para trabalhos futuros.

2 - O Problema de Recobrimento de Rotas (PRR)

2.1 - Introdução

A proposta deste capítulo é a de fornecer uma descrição concisa e ao mesmo tempo esclarecedora sobre o Problema de Recobrimento de Rotas (PRR), dando o embasamento necessário para o entendimento de nosso trabalho de tese e das dificuldades encontradas na resolução deste problema.

Inicialmente, faremos uma apresentação deste problema e em seguida, daremos uma visão geral dos principais trabalhos que têm sido desenvolvidos para a resolução do PRR e de seus problemas correlatos.

Concluindo nossa exposição, teremos uma apresentação detalhada das duas formulações de programação inteira e de um conjunto de regras de redução, propostas na literatura para a resolução do PRR. Observando que estas formulações e regras de redução serviram como base para o desenvolvimento de nosso trabalho de tese.

2.2 - Definição do Problema

Seja $G = (N, E)$ um grafo não direcionado, sendo N o conjunto de vértices formado pela união de dois conjuntos disjuntos V e W, com $N = V \cup W$, $V \cap W = \emptyset$ e $E = \{(v_i, v_j) \mid v_i, v_j \in N\}$ o conjunto de arestas. O conjunto V está associado aos vértices que podem ou não fazer parte da rota, ou seja, podem ou não ser visitados. T é um subconjunto de vértices de V que obrigatoriamente devem fazer parte da rota (devem ser visitados). W é o conjunto de vértices que devem ser cobertos pelos vértices do conjunto V que fazem parte da rota. A matriz de distâncias $D = [d_{ij}]$ está definida sobre E e satisfaz a desigualdade triangular $d_{ik} + d_{kj} \geq d_{ij}, \ \forall \ v_i, v_j, v_k \in N$. A cada aresta (v_i, v_j) de E associamos um valor d_{ij}, que representa a distância percorrida (comprimento) do vértice v_i para o vértice v_j nesta aresta (Gendreau[1]).

A partir destas considerações, temos que o Problema de Recobrimento de Rotas (figura 2.1) consiste em determinar uma rota de comprimento mínimo ou um ciclo Hamiltoniano (Swarficter[2]) sobre um subconjunto de V, de forma que todos os vértices de T estejam na rota e todos os vértices de W sejam cobertos por algum vértice desta rota. Um vértice $w_l \in W$ é dito coberto se existir pelo menos um vértice $t_k \in T$ ou um vértice $v_j \in V \backslash T$, tal que $d_{lk} \leq d$ ou $d_{lj} \leq d$, sendo d distância máxima de cobertura. Um vértice $v_j \in V \backslash T$ estará na rota se ele for necessário para cobrir algum vértice w_l do conjunto W (Gendreau[1]).

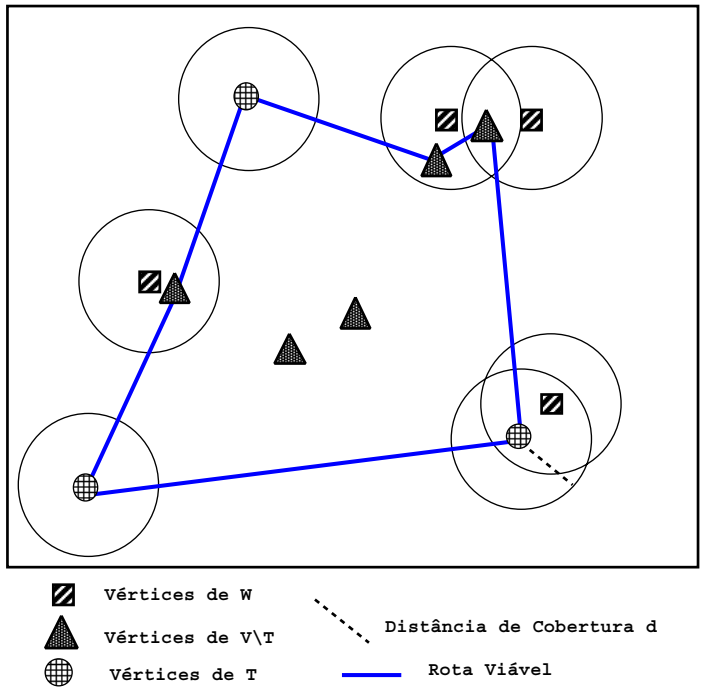

Figura 2.1 - Exemplo de uma rota viável do Problema de Recobrimento de Rotas

Observando a figura 2.1, temos o exemplo de uma rota viável para o PRR onde todos os vértices do conjunto W estão cobertos. Isto é, para cada vértice $w_l \in W$, existe pelo menos um vértice $v_j \in V\backslash T$ ou um vértice $t_k \in T$ a uma distância de no máximo d do vértice w_l e todos os vértices t_k do conjunto T fazem parte da rota.

Dentro desse escopo, podemos caracterizar o Problema do Caixeiro Viajante (PCV)(Wolsey[3], Lawler *et al.*[4]) como um caso particular do PRR, quando a distância de cobertura é nula, $d = 0$, e quando o conjunto W é igual ao conjunto $V\backslash T$ ($W = V\backslash T$).

Existem diversas aplicações associadas ao Problema de Recobrimentos de Rotas, dentre as quais podemos destacar:

• Planejamento de serviços médicos em áreas rurais, onde apenas um subconjunto de áreas receberão estes serviços diretamente (presença de uma equipe médica) e as demais áreas deverão estar em uma vizinhança (próximas) dessas cidades, com a finalidade de acessarem também os serviços médicos.

• O Transporte em Dois Níveis, onde apenas as cidades principais fazem parte da rota, isto é, serão diretamente atendidas, por exemplo, por carretas (1^o nível) e as demais cidades serão atendidas por outros tipos de transporte (2^o nível) que estarão a uma distância máxima pré-definida das cidades principais (1^o nível).

• A escala de aeronaves e tripulações, onde o planejamento da rota não inclui a visita em todas as cidades diretamente. Portanto, é necessário que a escala inclua paradas em cidades que estejam a uma distância máxima (distância de cobertura pré-definida) das cidades que não fazem parte da rota, facilitando a ligação entre estas cidades.

• Localização de postos de coleta dos correios em uma cidade.

Existem inúmeras outras aplicações que podem ser formuladas como o Problema de Recobrimento de Rotas, onde os locais ou cidades que não fazem parte da rota devem estar a uma distância máxima pré-definida dos locais que serão diretamente atendidos pela rota.

2.3 - Panorama do Problema

Nesta seção, faremos uma exposição dos principais trabalhos que vêm sendo desenvolvidos para resolução do Problema de Recobrimento de Rotas (PRR) e para os seus problemas correlatos.

O Problema de Recobrimento de Rotas, que foi abordado primeiramente por Current[5], em 1981, em sua tese de doutorado e referido na literatura por *The Covering Tour Problem (CTP)*, é uma generalização do Problema do Caixeiro Viajante (PCV)(Lawler[4]).

Em seu trabalho de 1995, Gendreau[1] propos uma nova formulação matemática para o PRR. Para a resolução desta formulação, foi desenvolvido um algoritmo exato baseado no método *Branch-and-Cut* (Wolsey[3]). Além disso, com a finalidade de produzir um limite superior (associado a uma solução viável) para este algoritmo, foi implementada uma heurística para o PRR que combina a heurística *GENIUS* (Gendreau[6]) desenvolvida para o PCV e o PRIMAL1 (Balas e Ho [7]) para o *Set Covering Problem (SCP)*.

A heurística *GENIUS* para o PCV, consiste em duas fases: A primeira fase, chamada *GENI* (*Generalized Insertion*), é iniciada com a seleção arbitrária de três vértices para formar uma rota inicial. A partir dessa rota inicial, um novo vértice é inserido a cada passo até que todos os vértices façam parte da rota.

Uma diferença em relação aos métodos que trabalham com outras formas de inserção de vértices é que nesse método as inserções não necessariamente ocorrem entre dois vértices consecutivos da rota parcial (em construção). Após esta fase de inserção, quando uma rota (solução) completa é obtida, tem início a fase de reotimização, conhecida como *US* (*Unstringing and Stringing*). Nesta fase, cada vértice é removido e reinserido na rota, usando o mesmo princípio da fase de inserção.

A outra heurística utilizada por Gendreau[1], chamada de PRIMAL1 (Balas e Ho [7]), inclui gradualmente na solução, vértices v_k (colunas) de acordo com um critério "guloso ", para encontrar o mínimo da função $f(c_k, b_k)$. Onde b_k é o número de vértices $w_l \in W$ cobertos pelo vértice v_k que satisfazem a desigualdade $d_{lk} \leq d$, mas que ainda não estão cobertos pela rota parcial, e c_k é o custo de inserção do vértice $v_k \in V$ na rota. Para este processo de minimização três tipos de funções $f(c_k, b_k)$ foram sugeridas:

$$f(c_k, b_k) = c_k / log_2\ (b_k)$$
$$f(c_k, b_k) = c_k / b_k$$
$$f(c_k, b_k) = c_k$$

Maniezzo *et al.*[8] propos uma solução para o PRR através da aplicação de três algoritmos baseados na metaheurística *Scatter Search* (Glover[9]). A idéia básica dessa metaheurística é a utilização de um conjunto de soluções R , chamado de Conjunto de Referência. Operadores de recombinação são aplicados nos elementos do Conjunto Referência, gerando novas soluções que posteriormente serão utilizadas como pontos de partida para procedimentos de busca local (Reeves[10]). As melhores soluções obtidas ao final deste processo são inseridas no próprio Conjunto de Referência.

Neste mesmo trabalho, são propostas regras de redução para o PRR e uma nova formulação matemática baseada na técnica de *Two-Commodity* (Finke[11]).

Em tal formulação temos a associação de duas variáveis de fluxo x_{ij} e x_{ji} para cada aresta $(v_i, v_j) \in E$. Sendo x_{ij} o número de vértices que podem ser visitados na rota e x_{ji} representa o número de vértices que já foram visitados na rota quando se está na aresta (v_i, v_j) . Um circuito é definido pelas variáveis de fluxo x_{ij} que representam o número de vértices que podem ser visitados, enquanto o segundo circuito é definido pelas variáveis de fluxo x_{ji} que representam o número de vértices já visitados x_{ji} .

Dos problemas de otimização similares ao PRR encontrados na literatura, podemos destacar:

- *The Covering Salesman Problem* (Current e Schilling[12]). A proposta deste problema é identificar uma rota de custo mínimo sobre um subconjunto de n cidades ($n \subset N$), tal que as demais cidades que não estão na rota estejam a uma distância máxima pré-definida de uma cidade que esteja na rota.

Neste trabalho os autores apresentaram uma formulação de programação inteira ($0-1$) e uma formulação multiobjetivo. Nesta formulação, além de minimizar o custo com a distância percorrida pela rota, deve-se minimizar o custo associado a utilização dos vértices do conjunto V . Além dessas formulações, desenvolveram uma heurística que fora utilizada para problemas convencionais, tais como: o Problema de Recobrimento e o Problema do Caixeiro Viajante.

• *The Shortest Covering Path Problem (SCPP)* apresentado no trabalho de Current *et al.*[13] em 1994, que consiste em identificar um caminho de custo mínimo, partindo de um vértice origem v_s e chegando em um vértice destino v_t ($v_s \neq v_t$), ambos previamente estabelecidos. Este caminho deve cobrir todos os vértices do grafo associado ao problema. Ou seja, todos os vértices devem estar a uma distância máxima de d (distância de cobertura) de pelo menos um dos vértices que fazem parte do caminho.

• *The Prize Collecting Traveling Salesman Problem (PCTSP)* abordado nos trabalhos de Balas[14] e Fischetti e Toth[15] e *The Selective Traveling Salesman Problem (STSP)* (Laporte e Martello[16]).

Nestes problemas existe um prêmio não negativo (peso) associado a cada vértice $v_i \in V$. E consistem ambos, em determinar uma rota passando pelo vértice v_1 e por um subconjunto de vértices do conjunto $V \backslash \{v_1\}$. O PCTSP consiste em minimizar o tamanho da rota, tal que a soma dos prêmios associados a cada vértice seja no mínimo igual a um certo valor pré-determinado p . O STSP consiste em maximizar o total de prêmios coletados, considerando que o comprimento da rota não ultrapasse um valor pré-determinado q .

• *The Median Tour Problem (MTP)* e *The Maximal Covering Tour Problem (MCTP)* (Current, Schilling[17]), assemelham-se em muitos pontos. Nesses dois problemas, a rota deve conter somente p dos n vértices do grafo, ($p \leq n$). Adicionalmente, ambos buscam a minimização do comprimento total da rota e também objetivam a maximização do acesso à rota por parte dos vértices não pertencentes a mesma.

A diferença básica entre esses dois problemas é que no primeiro deseja-se minimizar o custo total da distância de acesso à rota por parte dos vértices que não fazem parte da mesma, enquanto no segundo, deseja-se minimizar a demanda total dos vértices não cobertos pelos vértices da rota.

Podemos destacar também alguns problemas que são variações do PRR original, onde um conjunto de restrições extras é inserido na formulação do PRR para satisfazer objetivos específicos de cada problema:

• O Problema de Recobrimento com Múltiplas Rotas (m-PRR) consiste em

determinar m rotas de comprimento mínimo, todas partindo e chegando a um mesmo vértice origem (Hachicha *et al.*[18]). Nesse problema são consideradas duas restrições particulares: (1) O número de vértices de cada rota não deve exceder um valor pré-fixado p e o comprimento de cada rota não deve exceder a um valor pré-fixado q . (2) Cada vértice $t_k \in T$ pertence a exatamente uma rota e cada vértice $w_l \in W$ deve ser coberto por pelo menos uma rota.

O m-PRR se reduz ao PRR quando m for igual a 1, o valor pré-fixado p for grande o suficiente para conter, no pior caso, todos os vértices de V e quando q for maior que o somatório do comprimento de todas as arestas associadas aos vértices do conjunto V .

• O Problema de Recobrimento de Rotas Generalizado (PRRG) (Motta[19]), no qual é permitido que o vértice $w_l \in W$ pertença à rota, caso este vértice seja necessário cobrir algum outro vértice do conjunto W ou a si próprio.

• O Problema de Recobrimento de Rotas com Time-Windows (PRRTW) (Des rochers *et al.*[20], Solomon[21] e Solomon *et al.*[22]), onde os vértices do conjunto V possuem um intervalo de tempo para serem visitados. Seja (e_i, l_i) , a janela de tempo (Time-Windows) do vértice $v_i \in V$, onde e_i é o tempo mínimo para visitar o vértice v_i e l_i é o tempo máximo para visitar o vértice v_i .

Neste problema associa-se a cada aresta (v_i, v_j) , além da distância d_{ij} , que representa a distância entre os vértices v_i e v_j , o tempo t_{ij} que representa o tempo gasto para ir do vértice v_i para o vértice v_j , com $v_i, v_j \in V$. Para cada vértice v_i do conjunto V determina-se o tempo em que a rota visitou aquele vértice, tempo este definido por p_i . O valor p_i tem que estar compreendido no intervalo de tempo permitido para a visita deste vértice v_i , ou seja, $e_i \leq p_i \leq l_i$.

Sendo (v_i, v_j) uma aresta pertencente à rota, p_i o tempo em que a rota visitou o vértice v_i e t_{ij} o tempo gasto na rota para ir do vértice v_i ao vértice v_j . Podemos então, definir o tempo em que a rota visitou o vértice v_j utilizando a aresta (v_i, v_j) por $p_j = p_i + t_{ij}$, de forma que a seguinte desigualdade seja respeitada:

$$e_j \leq p_j = p_i + t_{ij} \leq l_j$$

Desigualdade esta referente à janela de tempo do vértice v_j , considerando o tempo do vértice origem como $p_o = 0$.

Concluindo nossa exposição, podemos citar o Problema de Recobrimento de Rotas em Clusters (PRRC) (Chisman[23] e Anily *et al.*[24]). Neste problema, os vértices dos conjuntos V e W são distribuídos em clusters e resolve-se o PRR em cada cluster, levando em conta a ordem em que estes clusters são definidos.

2.4 - Formulações Existentes para o PRR

Nessa seção, apresentamos as duas formulações de programação inteira propostas respectivamente por Gendreau *et al.* [1] e Maniezzo *et al.* [8] para o Problema de Recobrimento de Rotas (PRR).

2.4.1 - Formulação de Gendreau, Laporte e Semet

• Seja $G = (N, E)$ um grafo não direcionado, sendo N o conjunto de vértices formado pela união dos conjuntos V e W, $N = V \cup W$, $V \cap W = \emptyset$ e $E = \{(v_i, v_j) \mid v_i, v_j \in N, i < j\}$ o conjunto de arestas.

• Seja y_k uma variável binária associada a cada vértice $v_k \in V$, cujo valor é igual a 1 se e somente se o vértice v_k pertencer à rota, e zero caso contrário.

• Seja d_{ij} a distância percorrida entre v_i e v_j na aresta (v_i, v_j), ($v_i, v_j \in N, i < j$).

• Para cada par de vértices $v_i, v_j \in V, i < j$, definimos x_{ij} igual a 1, se e somente se, a aresta (v_i, v_j) pertencer à rota e zero, caso contrário.

• Se $t_k \in T$, então $y_k = 1$.

• O conjunto S_l, é formado por todos os vértices $v_k \in V$ que cobrem $w_l \in W$ ($\forall\, w_l \in W$, $S_l = \{v_k \in V \mid d_{kl} \leq d\}$), sendo d a distância de cobertura e d_{kl} a distância entre os vértices v_k e w_l).

• S é um subconjunto qualquer do próprio conjunto V, tal que $T \backslash S \neq \emptyset$.

$$Minimizar \quad \sum_{i<j} d_{ij}.x_{ij} \qquad (2.1)$$

$$Sujeito \quad a$$

$$\sum_{v_k \in S_l} y_k \ \geq \ 1 \quad (\forall \ w_l \in W) \qquad (2.2)$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} \ = \ 2y_k \quad (\forall \ v_i, v_j, v_k \in V) \qquad (2.3)$$

$$\sum_{v_i \in S, \ v_j \in V \backslash S \quad ou \quad v_j \in S, \ v_i \in V \backslash S} x_{ij} \ \geq \ 2y_k \qquad (2.4)$$

$$(S \subset V, \ 2 \leq |S| \leq n-2, \ T \backslash S \neq \emptyset, v_k \in S)$$

$$y_k = 1 \quad (\forall \ t_k \in T) \qquad (2.5)$$

$$y_k \in \{0,1\} \quad (\forall \ v_k \in V \backslash T) \qquad (2.6)$$

$$x_{ij} \in \{0,1\} \quad (\forall \ v_i, \ v_j \in V, \ i<j) \qquad (2.7)$$

A formulação definida por (2.1)-(2.7) é tal que:

As restrições (2.2) garantem que cada vértice $w_l \in W$ seja coberto por no mínimo um vértice $v_k \in V$. As restrições (2.3) asseguram que o número de arestas incidentes ao vértice $v_k \in V$ seja igual 2 se o vértice v_k fizer parte da rota, e zero caso contrário. As restrições (2.4) asseguram a conectividade da rota, não permitindo ciclos desconexos com a origem. As restrições (2.5) garantem que cada vértice $t_k \in T$ necessariamente faça parte da rota. Finalmente, as restrições (2.6) e (2.7) representam as condições de integralidade do problema.

2.4.2 - Formulação de Maniezzo, Baldacci, Boschetti e Zamboni

Nesta formulação (descrita por (2.8) a (2.16)) temos a associação de duas variáveis de fluxo x_{ij} e x_{ji} a cada aresta (v_i, v_j) do grafo $G = (N, E)$.

• A rota parte de um vértice origem t_0 , onde $t_0 \in T$.

• Seja $E' = \{(v_i, v_j) : v_i,\ v_j \in N,\ i < j\}$.

• Se a rota vai do vértice v_i ao vértice v_j , então x_{ij} representa o número de vértices que podem ser visitados e x_{ji} representa o número de vértices visitados até o momento.

• Seja ϵ_{ij} uma variável binária que assume valor 1, se e somente se, a aresta $(v_i, v_j) \in E'$ é considerada na rota e zero, caso contrário.

• Seja y_k uma variável binária que assume valor 1, se e somente se, o vértice $v_k \in V$ for visitado e zero, caso contrário.

$$Minimizar \qquad \sum_{(v_i,v_j)\in E'} d_{ij}.\epsilon_{ij} \qquad (2.8)$$

$$Sujeito \quad a$$

$$\sum_{v_j \in V} x_{ij} - \sum_{v_j \in V} x_{ji} \ = \ -2y_i \qquad (2.9)$$

$$(v_i \in V, v_i \neq t_0\ ,\ v_j \neq t_0\ , t_0\ \in T)$$

$$\sum_{v_j \in V} (x_{ij} + x_{ji}) \ = \ 2y_i(|V|-1) \quad (v_i \in V) \qquad (2.10)$$

$$\sum_{v_k \in S_l} y_k \ \geq \ 1 \quad (S_l = \{v_k \in V \mid d_{kl} \leq d\}, w_l \in W) \qquad (2.11)$$

$$x_{ij} + x_{ji} \ = \ (|V|-1)\epsilon_{ij} \quad (v_i, v_j) \in E' \qquad (2.12)$$

$$x_{ij} \ \geq \ 0 \quad v_i, v_j \in V, \quad v_i \neq v_j \qquad (2.13)$$

$$y_k \;=\; 1 \quad (\forall\, t_k \in T) \tag{2.14}$$

$$y_k \in \{0,1\} \;\; (\forall\, v_k \in V \backslash T) \tag{2.15}$$

$$\epsilon_{ij} \in \{0,1\} \;\; (\forall\, (v_i, v_j) \in E') \tag{2.16}$$

As restrições (2.9) e (2.13) asseguram o fluxo viável entre os vértices v_i e v_j. As restrições (2.10) e (2.12) forçam que o grau do vértice v_k pertencente à rota seja igual a 2. As restrições (2.11) garantem a cobertura de cada vértice do conjunto W e as restrições (2.14) garantem que todos os vértices do conjunto T estejam na rota. As restrições (2.15) e (2.16) definem as condições de integralidade do problema.

Na figura 2.2 apresentada abaixo, temos o exemplo de uma rota com 9 vértices, representada pelas variáveis de fluxo x_{ij} e x_{ji}.

Os arcos vermelhos (associados as variáveis x_{ij}) representam o número de vértices que podem ser visitados e os arcos tracejados (associados as variáveis x_{ji}) representam o número de vértices já visitados.

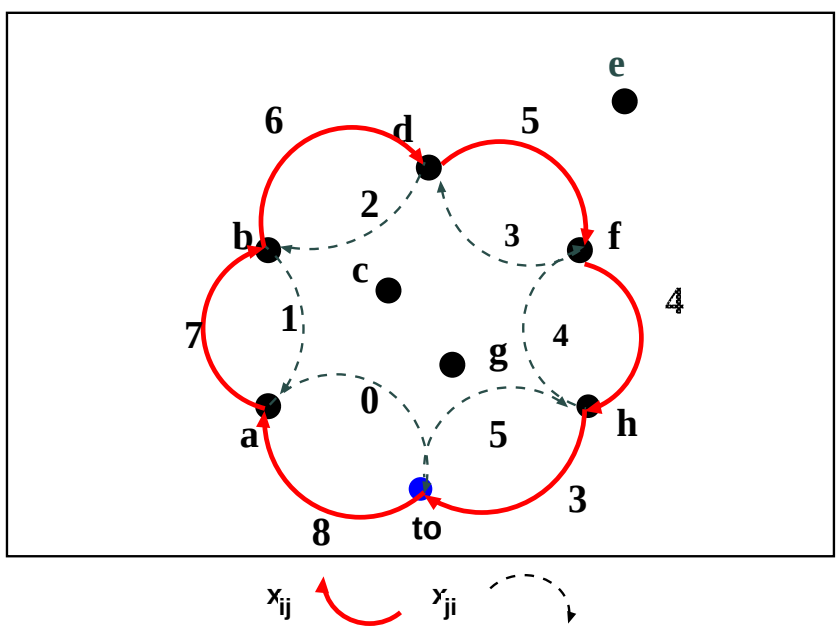

Figura 2.2 - Ilustração da Formulação de Maniezzo

2.5 - Regras de Redução para o PRR

Considerando as formulações apresentadas para o PRR na seção anterior, podemos observar, que dependendo do número de vértices do grafo G associado ao PRR, essas formulações podem ser pouco eficientes. Ou seja, teremos um razoável número de variáveis inteiras e restrições, o que pode tornar difícil a obtenção de uma solução ótima, ou até mesmo viável, em tempo computacionalmente razoável quando aplicamos algum método exato.

Como uma alternativa para tentar reduzir esta dificuldade, tem-se incorporado aos algoritmos exatos e aproximados as regras de redução. O objetivo dessas regras é reduzir o número de vértices do grafo original e conseqüentemente o número de variáveis e restrições da formulação, de forma que o conjunto de soluções associadas ao grafo reduzido, grafo obtido a partir do grafo original após aplicar a regra de redução, seja um subconjunto de soluções viáveis para o grafo original.

Ao se reduzir um grafo, deve-se ter a garantia que nenhuma solução ótima associada a esse grafo seja descartada. Portanto, ao se aplicar as regras de redução deve-se retirar somente os vértices que não formaram parte em nenhuma solução ótima.

No trabalho de Maniezzo *et al.*[8] encontramos 4 regras de redução aplicadas para resolução do PRR, cuja descrição e exemplificação veremos a seguir.

Seja δ_{li} uma variável binária que assume valor 1 se $d_{li} \leq d$ (d distância máxima pré-estabelecida, d_{li} distância entre os vértices $w_l \in W$ e $v_i \in V$) e 0 caso contrário.

REGRA 1) Remover o vértice $w_l \in W$ do grafo original, se $\delta_{li} = 1$, $\forall v_i \in V \backslash T$.

REGRA 2) Remover o vértice $w_l \in W$ do grafo original, se existe $w_m \in W$, $w_l \neq w_m$ tal que $\delta_{li} \leq \delta_{mi}$, $\forall v_i \in V \backslash T$.

REGRA 3) Remover o vértice $w_l \in W$ do grafo original, se existe $t_i \in T$, tal que $\delta_{li} = 1$.

REGRA 4) Remover o vértice $v_i \in V \backslash T$ do grafo original, se $\delta_{li} = 0$, $\forall w_l \in W$.

Observamos, que em alguns casos, após aplicarmos a regra 1 ou a regra 2, temos grafos reduzidos cujas rotas são inviáveis para o grafo original. Isto é, são retirados vértices do grafo original sem a garantia de que as restrições do PRR sejam satisfeitas.

Na figura 2.3, temos um exemplo onde a regra 1 falha. Considere o grafo G com $|T| = 3$, $|W| = 3$ e $|V \backslash T| = 3$. Podemos observar que qualquer rota viável para esse grafo (figura 2.3(a)) deverá conter pelo menos um dos vértices $v_i \in V \backslash T$, pois estes vértices cobrem todos os vértices $w_l \in W$. Desta forma, a condição de cobertura de cada vértice $w_l \in W$ é garantida por um dos vértices v_i do conjunto $V \backslash T$.

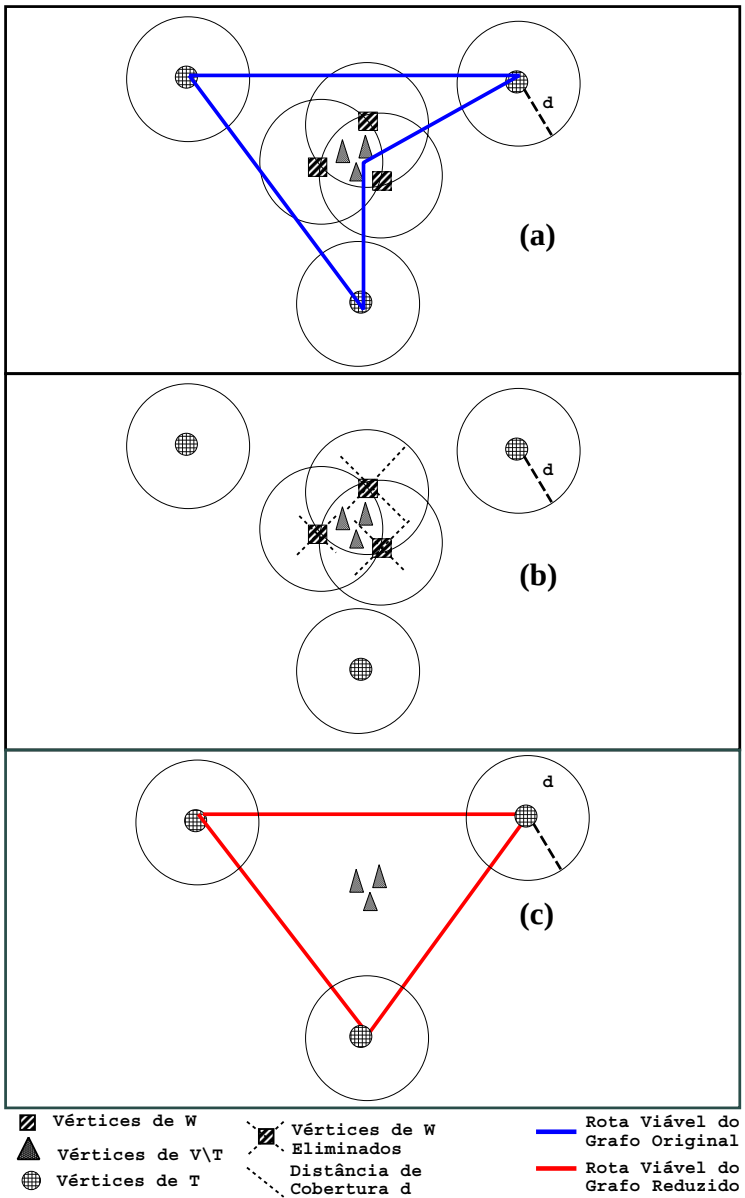

Figura 2.3 - Exemplo de aplicação da regra de redução 1

A regra 1 descrita na literatura, retira o vértice $w_l \in W$ do grafo original, se todos os vértices do conjunto $V \backslash T$ cobrem o vértice w_l . Portanto, aplicando a regra 1 no grafo da figura 2.3(a) eliminaremos cada vértice $w_l \in W$ resultando no grafo reduzido da figura 2.3(b). Neste grafo reduzido, os vértices do conjunto $V \backslash T$ não estão cobrindo nenhum vértice do conjunto de W (todos estes vértices foram eliminados), portanto não farão parte de nenhuma rota viável.

Na figura 2.3(c) apresentamos uma rota viável para o grafo reduzido que não é viável para o grafo original, pois os vértices $w_l \in W$ não são cobertos por esta. Conseqüentemente, aplicando a regra 1, podemos obter rotas não viáveis para o grafo original.

É imprescindível que qualquer rota viável associada ao grafo reduzido seja viável para o grafo original no qual foi aplicada a regra de redução. As regras de redução simplificam o conjunto de soluções do grafo original sem desconsiderar qualquer rota viável que seja candidata à rota ótima.

A regra 2 elimina o vértice $w_l \in W$, se todos os vértices $v_i \in V \backslash T$ que o cobrirem, também cobrirem um outro vértice $w_m \in W$, $w_m \neq w_l$. Na figura 2.4(a) temos um exemplo de um grafo no qual poderemos aplicar a regra 2.

Considere o grafo G com $|T| = 1$, $|W| = 3$ e $|V \backslash T| = 5$. Podemos observar que qualquer rota viável para esse grafo (figura 2.4(a)), deverá conter um dos vértices $v_i \in V \backslash T$ que cobrem w_l , garantindo assim a viabilidade do problema. Portanto, aplicando a regra 2 no grafo da figura 2.4(a), eliminamos o vértice $w_l \in W$, resultando no grafo reduzido da figura 2.4(b).

Na figura 2.4(c), temos uma rota viável para esse grafo reduzido, que não é viável para o grafo original, pois o vértice $w_l \in W$ não está coberto por esta rota, causando assim uma inviabilidade para este problema. Para o grafo original considerado (figura 2.4(a)), qualquer rota viável deve cobrir os vértices w_l, w_m do conjunto W . Portanto, será necessário que um dos vértices $v_i \in V \backslash T$ que cobrem o vértice $w_l \in W$ façam parte de qualquer rota viável.

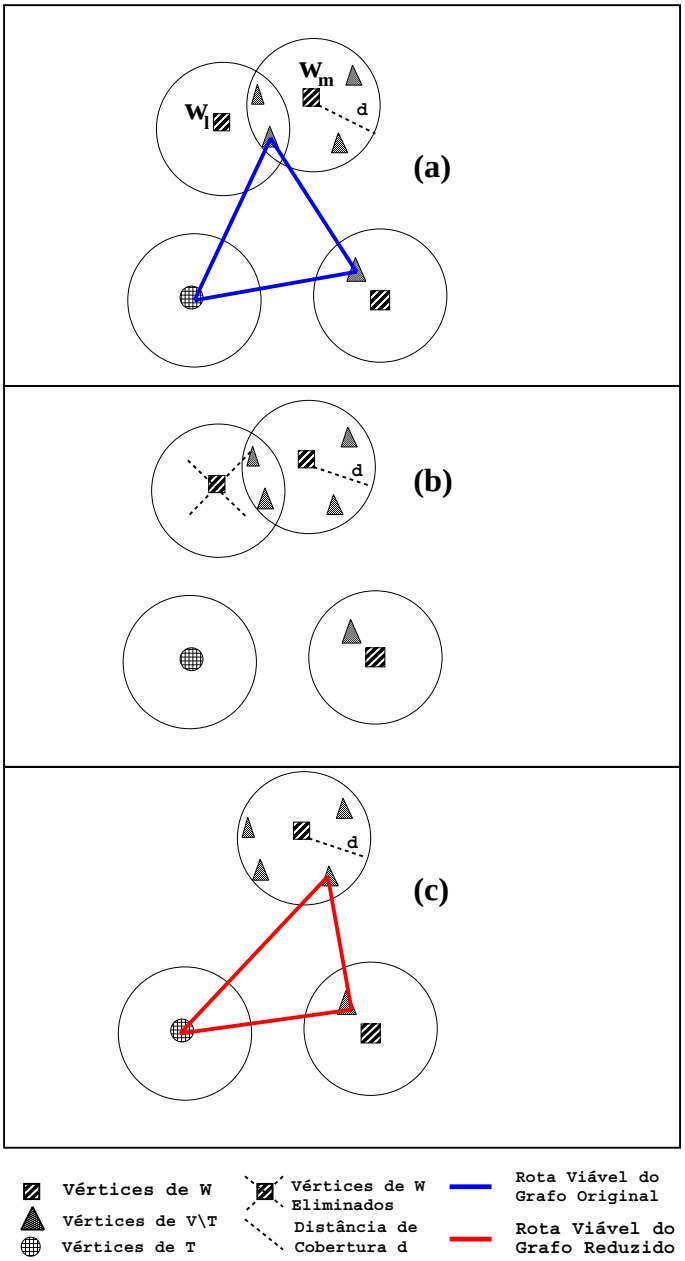

Figura 2.4 - Exemplo de aplicação da regra 2

A regra 3 elimina um vértice w_l do conjunto W se este vértice for coberto por um vértice t_i do conjunto T, ou seja, $d_{li} \leq d$. Todos os vértices do conjunto T deverão fazer parte de qualquer rota viável e qualquer vértice w_l do conjunto W que esteja a uma distância de no máximo d destes vértices do conjunto T será coberto.

Na figura 2.5 temos um exemplo de aplicação da regra 3, considerando um grafo G com $|T| = 3$, $|W| = 3$ e $|V \backslash T| = 2$. Na figura 2.5(a) temos uma rota viável para o grafo original, pois observamos a cobertura de todos os vértices do conjunto W e a condição de que todos os vértices do conjunto T estejam na rota. Aplicando a regra 3 na figura 2.5(a), retiramos os vértices w_l do conjunto W que estão sendo cobertos por algum vértice t_i do conjunto T, resultando no grafo reduzido da figura 2.5(b). E finalmente, no grafo reduzido apresentado na figura 2.5(c), temos a representação de uma rota viável. Observamos que qualquer rota viável para o grafo reduzido é viável para o grafo original da figura 2.5(a).

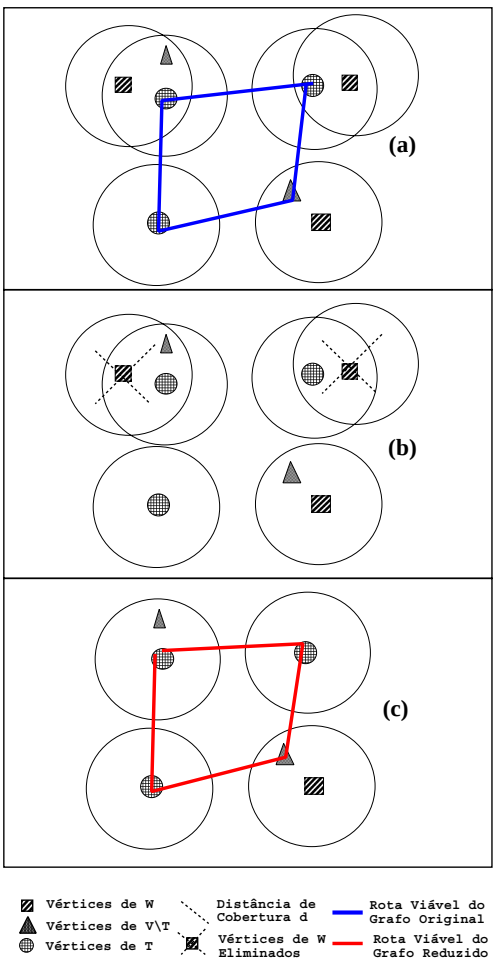

Figura 2.5 - Exemplo de aplicação da regra 3

A regra 4 elimina os vértices do conjunto $V\backslash T$ que não cobrem nenhum vértice do conjunto W. Lembramos que um vértice do conjunto $V\backslash T$ fará parte da rota se o mesmo for necessário para cobrir algum vértice $w_l \in W$. Vejamos o grafo original com uma rota viável, figura 2.6(a), com $|T| = 1$, $|W| = 2$ e $|V\backslash T| = 7$. Aplicando a regra 4, obtemos o grafo reduzido que está representado na figura 2.6(b) e que mantém apenas os vértices do conjunto $V\backslash T$ que realmente cobrem algum vértice do conjunto W (figura 2.6(b)). A rota viável para o grafo reduzido (figura 2.6(c)) é também viável para o grafo original. Observamos que qualquer rota viável para o grafo reduzido (figura 2.6(c)) também é viável para o grafo original da figura 2.6(a).

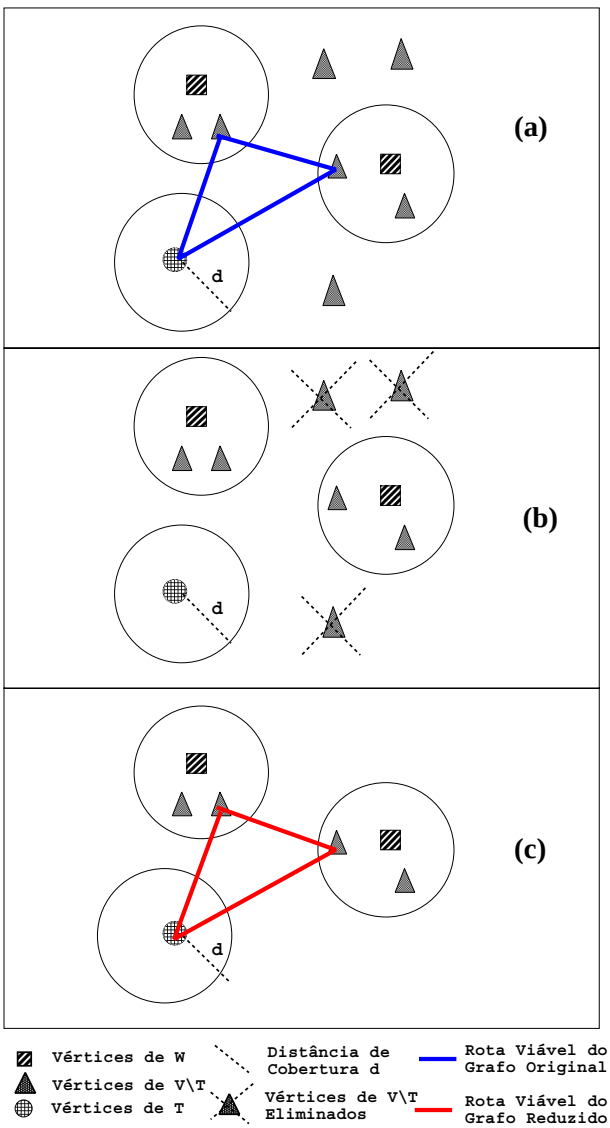

Figura 2.6 - Exemplo de aplicação da regra 4

3 - Nova Formulação Matemática para o PRR

3.1 - Introdução

Conforme descrito no capítulo 2, o Problema de Recobrimento Rotas é um problema de otimização combinatória de difícil solução e que está associado a um vasto conjunto de aplicações reais.

Dependendo do número de vértices do grafo de entrada associado a uma instância do PRR, a ação e a eficiência dos métodos exatos pode ficar limitada, ainda que consideremos o progresso obtido nos últimos anos nesse setor. Em alguns casos, o problema pode ser insolúvel do ponto de vista exato (Maculan[25] e Wolsey[3]).

Desta forma, com a finalidade de tentar reduzir este tipo de dificuldade, apresentamos neste capítulo as seguintes propostas alternativas para a resolução do PRR:

- Uma nova formulação matemática que difere das existentes na literatura, por fazer uso de variáveis de fluxo para evitar ciclos desconexos da origem.

- A utilização da Relaxação Lagrangeana (Geoffrion[26]) considerando a nova formulação matemática proposta para o PRR. Com a utilização da relaxação, tem-se a expectativa de obter limites inferiores melhores do que aqueles obtidos mediante a aplicação da relaxação linear (Bazaraa[27]).

- A implementação de uma Heurística Lagrangeana considerando as particularidades do PRR. Com a utilização dessa heurística, "ajustamos " a solução obtida a partir da relaxação lagrangeana, de forma a produzir uma solução viável que constituirá um limite superior para a solução ótima associada ao PRR.

3.2 - Formulação Matemática

Nesta seção, apresentamos uma nova formulação matemática que se diferencia das demais formulações existentes na literatura quanto as restrições que asseguram a não formação de ciclos desconexos da origem. Estas restrições estão diretamente ligadas às variáveis de fluxo que associamos a cada aresta do grafo relacionado a uma instância do Problema de Recobrimento de Rotas.

Seja $G = (N, E)$ um grafo não direcionado, sendo N o conjunto de vértices formado pela união de dois conjuntos disjuntos V e W, com $N = V \cup W$, $V \cap W = \emptyset$ e $E = \{(v_i, v_j) \mid v_i, v_j \in N\}$ o conjunto de arestas. O conjunto V está associado aos vértices que podem ou não fazer parte da rota, ou seja, podem ou não ser visitados. T é um subconjunto de vértices de V que obrigatoriamente devem fazer parte da rota (devem ser visitados). W é o conjunto de vértices que devem ser cobertos pelos vértices que fazem parte da rota (vértices de V). A

matriz de distâncias $D = [d_{ij}]$ está definida sobre E e satisfaz a desigualdade triangular $d_{ik} + d_{kj} \geq d_{ij}$, $\forall\ v_i, v_j, v_k \in N$. A cada aresta (v_i, v_j) de E é associado um valor d_{ij} , que representa a distância percorrida (comprimento) do vértice v_i para o vértice v_j nesta aresta (Gendreau[1]).

- Seja y_k uma variável binária associada a cada vértice $v_k \in V$, cujo valor é igual a 1 se e somente se o vértice v_k pertencer à rota, e zero caso contrário.

- d_{ij} é a distância percorrida na rota do vértice v_i para o vértice v_j na aresta (v_i, v_j)

- Seja x_{ij} uma variável binária que assume o valor 1 se à aresta (v_i, v_j) pertence à rota, e zero caso contrário, ($v_i, v_j \in V$).

- $t_0 \in T$ é o vértice origem.

- Se $t_k \in T$, então $y_k = 1$.

- O conjunto S_l , é formado por todos os vértices $v_k \in V$ que cobrem $w_l \in W$, ($\forall\ w_l \in W$, $S_l = \{v_k \in V \mid d_{lk} \leq d\}$), d é a distância de cobertura e d_{lk} a distância percorrida do vértice $w_l \in W$ para o vértice $v_k \in V$.

- Seja z_{ij} uma variável não-negativa, associada a cada aresta $(v_i, v_j) \in E$ e que representa a quantidade de fluxo escoado do vértice v_i para o vértice v_j.

$$(PRR) \qquad Minimizar \qquad \sum_{\forall\ v_i, v_j \in V} d_{ij}.x_{ij} \tag{3.1}$$

$$Sujeito\ \ a$$

$$\sum_{v_k \in S_l} y_k \ \geq\ 1 \quad (\forall\ w_l \in W) \tag{3.2}$$

$$\sum_{v_i \neq v_k} x_{ik} + \sum_{v_j \neq v_k} x_{kj} = 2y_k \quad (\forall\ v_i, v_j, v_k \in V) \tag{3.3}$$

$$\sum_{v_j \neq v_k} z_{kj} = \sum_{v_i \neq v_k} z_{ik} \ +\ y_k \quad (\forall\ v_i,\ v_j,\ v_k \in V \backslash \{t_0\}) \tag{3.4}$$

$$\sum_{v_j \in\ V \backslash \{t_0\}} z_{0j} \ =\ 1 \tag{3.5}$$

$$\sum_{v_j \in V \setminus \{t_0\}} z_{j0} \;=\; \sum_{v_j \in V} y_j \tag{3.6}$$

$$x_{ij} \;\leq\; z_{ij} \quad (\forall\, v_i, v_j \in V) \tag{3.7}$$

$$x_{ij} \;\geq\; \frac{z_{ij}}{|V|+1} \quad (\forall\, v_i, v_j \in V) \tag{3.8}$$

$$|V| \;=\; \textit{número de vértices do conjunto } V$$

$$y_k \;=\; 1 \quad (\forall\, t_k \in T) \tag{3.9}$$

$$y_k \in \{0,1\} \;\; (\forall\, v_k \in V \setminus T) \tag{3.10}$$

$$x_{ij} \in \{0,1\} \;\; (\forall v_i, v_j \in V) \tag{3.11}$$

$$z_{ij} \;\geq\; 0 \;\; (\forall v_i, v_j \in V) \tag{3.12}$$

Nesta formulação, a função objetivo agrega a soma das distâncias entre todos vértices consecutivos da rota. As restrições (3.2) garantem que cada vértice $w_l \in W$ seja coberto por no mínimo um vértice $v_k \in V$ da rota. As restrições (3.3) asseguram que o número de arestas incidentes ao vértice $v_k \in V$ seja igual 2 se o vértice v_k for visitado pela rota, e zero caso contrário. As restrições (3.4)-(3.6) asseguram a conectividade da rota, não permitindo a formação de ciclos desconexos da origem. As restrições (3.7) e (3.8) mostram que a rota gerada a partir da variável de fluxo z_{ij} coincide com a rota gerada pela variável de decisão x_{ij} . As restrições (3.9) garantem que cada vértice de T necessariamente fará parte da rota. Finalmente, as restrições (3.10) e (3.11) representam as condições de integralidade do problema.

A proposição a seguir assegura que as retrições (3.4), (3.5) e (3.6) evitam a formação de ciclos desconexos da origem.

Proposição: Seja S uma rota viável para o PRR, gerada pela formulação definida pelas restrições de (3.2) a (3.12). Então a rota S é formada por um único ciclo contendo o vértice origem t_0.

Demonstração (por absurdo): Suponhamos que S seja uma rota viável para o PRR e que S contenha um ciclo C de tamanho $\alpha + 1$, desconexo da origem.

Seja C o ciclo definido pelos vértices, v_k, v_{k+1}, v_{k+2}, ..., $v_{k+\alpha}$, v_k, pertencentes ao conjunto V (fig.3.1).

Admitindo que a quantidade de fluxo escoado do vértice v_k para o vértice v_{k+1}, seja $z_{k\,k+1} = f \geq 0$ e usando a restrição (3.4), temos:

$$z_{k+1\,k+2} \;=\; z_{k\,k+1} \;+\; 1 \;=\; f \;+\; 1$$

Considerando novamente a restrição (3.4) e repetindo $(\alpha - 1)$ vezes o processo de substituição descrito acima, teremos:

$$z_{k+\alpha\,k} \;=\; f \;+\; \alpha \qquad (\alpha \;>\; 1\,)$$

$$(\Rightarrow)\; z_{k\,k+1} \;=\; z_{k+\alpha\,k} \;+\; 1 \;=\; f \;+\; \alpha \;+\; 1$$

Como pela hipótese $z_{k\,k+1} = f$, chegamos então a um absurdo, pois temos $f \;=\; f \;+\; \alpha + 1$. Desta forma, concluímos que se S é uma rota viável, então ela não pode conter ciclos desconexos da origem . □

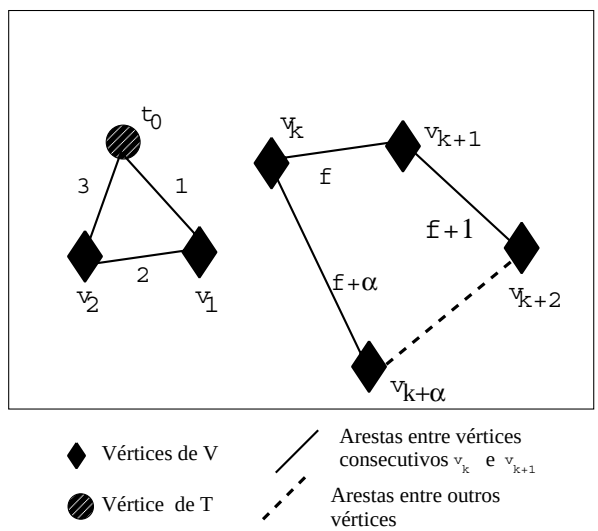

Figura 3.1 - Ilustração da Proposição

3.3 - Relaxação e Heurística Lagrangeana

3.3.1 - Introdução:

Tradicionalmente, a relaxação lagrangeana é uma técnica importante para gerar limites inferiores utilizados em algoritmos para resolver problemas de otimização combinatória (Geoffrion[26]). Com a aplicação dessa relaxação remove-se da formulação original grupos de restrições de modo a obter uma formulação menos complexa. As restrições "complicantes " são então convertidas em penalidades com multiplicadores de Lagrange (variáveis duais) e adicionadas à função objetivo.

Dentro desta perspectiva, apresentamos a seguir, a proposta de uma relaxação lagrangeana para a nova formulação desenvolvida para o PRR (vide seção 3.2). Nesta relaxação, agregamos à função objetivo as restrições $\sum_{v_k \in S_l} y_k \geq 1 \quad (\forall\, w_l \in W)$.

Além disto, apresentamos uma heurística lagrangeana que constrói uma solução viável para o PRR a partir da solução obtida mediante a aplicação da relaxação lagrangeana.

Inicialmente, com finalidade de facilitar o entendimento de nossa proposta, faremos uma breve descrição da técnica de relaxação lagrangeana e dando seqüência, apresentaremos a relaxação e a heurística lagrangeana implementadas para o PRR.

3.3.2 - Descrição da Relaxação Lagrangeana

Nossa exposição será desenvolvida para os problemas de programação linear inteira 0-1, mas podemos estender, sem dificuldades, os resultados apresentados para outros problemas de programação linear inteira.

Considere o seguinte problema:

$$(P) \quad Minimizar \;\; z = \;\; cx \qquad (3.13)$$

$$Sujeito\ a$$

$$Ax \geq b \qquad (3.14)$$

$$Bx \geq d \qquad (3.15)$$

$$x \in \{0, 1\} \qquad (3.16)$$

Um limite inferior para o problema (P) pode ser obtido via relaxação linear (Bazaraa [27]), considerando um problema de programação linear contínua $\overline{P}$ com as restrições de integralidade (3.16) relaxadas.

$$(\overline{P}) \quad Minimizar \;\; z = cx \qquad (3.17)$$

$$Sujeito\ a$$

$$Ax \geq b \qquad (3.18)$$

$$Bx \geq d \qquad (3.19)$$

$$0 \leq x \leq 1 \qquad (3.20)$$

O problema $(\overline{P})$ apresentado acima, pode ser resolvido de forma exata através da aplicação de um método de programação linear, como por exemplo, o método *Simplex* (Bazaraa[27]) ou o método de Pontos Interiores. A solução obtida a partir desta relaxação linear será um limite inferior para o problema (P).

Todavia, em muitos casos, resolver a relaxação linear de (P) definida por $(\overline{P})$ pode ser inadequada, tipicamente porque:

- $(\overline{P})$ Pode ter também um grande número de variáveis e/ou restrições.
- A qualidade do limite inferior obtido em $(\overline{P})$ pode ser ruim em muito casos.

Desta forma, necessitamos de técnicas alternativas para gerar limites inferiores para (P) .

Como uma técnica alternativa e bem utilizada para obtenção de limites inferiores para problemas com a estrutura de (P), podemos destacar a relaxação lagrangeana (Geoffrion[26]). Esta técnica, por exemplo, foi utilizada com sucesso por Held e Karp (1970)[28], na resolução do Problema do Caixeiro Viajante.

Definimos a relaxação lagrangeana de (P) com respeito ao conjunto de restrições $Ax \geq b$ pela introdução de um vetor de multiplicadores $(\lambda \geq 0)$, conhecido como vetor de multiplicadores de lagrange, que está relacionado a este conjunto de restrições e que inserido na função objetivo, fornece o problema descrito por (3.21)-(3.23):

$$(PL) \quad L(\lambda) \;=\; Minimizar \;\{\; cx \;+\; \lambda(b - Ax) \;\} \tag{3.21}$$

$$Sujeito\ a$$

$$Bx \geq d \tag{3.22}$$

$$x \;\in\; \{0, 1\} \tag{3.23}$$

Observamos que $L(\lambda)$ é uma cota inferior de $v(P)$ (valor ótimo de P), desde que $\lambda \geq 0$ e $(b - Ax) \leq 0$, ou seja:

$$L(\lambda) \leq v(P) \tag{3.24}$$

Visando então à busca da maior cota inferior possível, consideramos o seguinte problema lagrangeano dual:

$$(PLD) \quad Maximizar \;\; L(\lambda) \tag{3.25}$$

$$Sujeito \;\; a$$

$$\lambda \geq 0 \tag{3.26}$$

Idealmente, o valor ótimo de (PLD) deveria ser igual ao valor ótimo do problema original (P) , mas normalmente, os dois problemas não têm valores ótimos iguais. Existe, então um *gap* de dualidade, que é medido pela diferença entre os valores ótimos desses problemas.

Embora (P) e (PL) tenham a mesma natureza (sejam problemas de programação inteira), a razão para a utilização da relaxação lagrangeana é a exclusão das restrições que tornam o problema mais complicado de ser resolvido do ponto de vista computacional (Geoffrion[26]) e a possibilidade de se obter limites inferiores melhores do que aqueles fornecidos pela relaxação linear (Bazaraa[27]).

3.3.3 - Relaxação e Heurística Lagrangeana para o PRR

Conforme observado na seção anterior, com a aplicação da relaxação lagrangeana, substituímos a resolução de um problema (P) por um problema relaxado (PL). Isto é, a função objetivo de (PL) encapsula a função objetivo do problema (P) mais um conjunto de restrições previamente escolhidas de (P).

Considerando então a nova formulação proposta para o PRR e agregando à função objetivo as restrições do tipo $\sum_{v_k \in S_l} y_k \geq 1 \quad (\forall\ w_l \in W)$ obtemos a seguinte relaxação:

$$(PL) \quad Minimizar \sum_{\forall\ v_i, v_j \in V} d_{ij}.x_{ij} \ + \ (\lambda.(1 - \sum_{v_k\ \in\ S_l} y_k)) \qquad (3.27)$$

$$Sujeito\ \ a$$

$$\sum_{v_i \neq v_k} x_{ik} + \sum_{v_j \neq v_k} x_{kj} = 2y_k \quad (\forall\ v_i, v_j, v_k\ \in\ V) \qquad (3.28)$$

$$\sum_{v_j \neq v_k} z_{kj} = \sum_{v_i \neq v_k} z_{ik} + y_k \quad (\forall\ v_i,\ v_j,\ \ v_k\ \in V \backslash \{t_0\}) \qquad (3.29)$$

$$\sum_{v_j \in V \backslash \{t_0\}} z_{t_0 j} = 1 \qquad (3.30)$$

$$\sum_{v_j\ \in\ V\ \backslash \{t_0\}} z_{j t_0} = \sum_{v_j\ \in\ V} y_j \qquad (3.31)$$

$$x_{ij} \leq z_{ij} \quad (\forall\ \ v_i, v_j \in V) \qquad (3.32)$$

$$x_{ij} \geq \frac{z_{ij}}{|V| + 1} \quad (\forall\ \ v_i, v_j \in V) \qquad (3.33)$$

$$|V| \ = \ número\ de\ vértices\ do\ conjunto\ V$$

$$y_k = 1 \quad (\forall\ t_k \in T) \qquad (3.34)$$

$$y_k \in \{0,1\} \quad (\forall\ v_k\ \in\ V \backslash T\ ) \tag{3.35}$$

$$x_{ij} \in \{0,1\} \quad (\forall\ v_i\ v_j\ \in\ V) \tag{3.36}$$

$$z_{ij}\ \geq\ 0\ \ (\forall v_i, v_j \in V) \tag{3.37}$$

Então, definimos o problema (PL) relaxando as restrições (3.2), que garantem a cobertura de cada vértice do conjunto W por pelo menos um vértice do conjunto V .

Em seguida, de forma a obter a melhor cota inferior para (PL), resolvemos o seguinte problema:

$$(PLD) \quad Maximizar\ \ L(\lambda) \tag{3.38}$$

$$Sujeito\ \ a \quad \lambda \geq 0$$

Para resolução do problema (PLD) , utilizamos o método do Subgradiente, um algoritmo que a partir de um conjunto inicial de multiplicadores λ_i , gera novos multiplicadores, num processo sistemático de ajuste dos mesmos (Geoffrion[26]).

Mesmo com a relaxação das restrições do tipo $\sum_{v_k \in S_l}\ y_k\ \geq\ 1\ (\forall\ w_l\ \in\ W)$, preservamos as restrições que garantem a conectividade da rota e que evitam formação de ciclos desconexos da origem. Além disso, garantimos que todos os vértices do conjunto T estão na rota.

Além da obtenção de soluções (x^*, λ^*) que fornecem limites inferiores para o problema original, desenvolvemos uma heurística lagrangeana simples, que a partir de uma solução x^* (obtida da relaxação), constrói uma solução x^{**} viável para o problema original.

A construção de x^{**} se dá a partir da inserção de vértices $v_i\ \in\ V \backslash T$ que inicialmente não tomam parte na rota (solução x^*) e que garantem a cobertura de todos os vértices $w_k\ \in\ W$ que não são cobertos por esta rota, devido à relaxação das restrições apresentadas acima.

Então, após a aplicação desta heurística, teremos uma solução modificada x^{**} , que será viável para o problema original e que poderá ser utilizada como um limite superior durante a aplicação do algoritmo do subgradiente.

Nas figuras 3.2 e 3.3, apresentadas a seguir, temos respectivamente o algoritmo do subgradiente e a heurística lagrangeana, utilizados na resolução da relaxação lagrangeana proposta para a resolução do PRR.

1) Faça $\pi = 2$ e $j = 0$

2) Faça $z_{UB} = +\infty$ (Limite Superior)

3) Faça $z_{LB} = -\infty$ (Limite Inferior)

4) Faça $dif_{Z_{UB}Z_{LB}} = 100$

5) Defina o $desvio$ (Diferença percentual entre z_{UB} e z_{LB})

6) Defina tot_{iter} (total de iterações do algoritmo)

7) Defina valores iniciais para os multiplicadores , λ_i , $i = 1, ..., |W|$

8) Enquanto ($j \leq tot_{iter}$) e ($dif_{Z_{UB}Z_{LB}} > desvio$) Faça

9) Faça $j = j + 1$

10) Resolva (PL) com valores atuais de λ_i , $i = 1, ..., |W|$
e obtenha uma solução x^* com valor z_{LB} (limite inferior)

11) Aplique Heurística-Lagrangeana(x^*) (Retorna z_{UB})

12) Faça $dif_{Z_{UB}Z_{LB}} = 100 * \frac{z_{UB} - z_{LB}}{z_{LB}}$

13) Para $i = 1$ até $|W|$ Faça $G_i = 1 - \sum_{v_k \in S_i} y_k$ Fim-Para

14) Faça $T = \frac{\pi(z_{UB} - z_{LB})}{\sum_{i=1}^{|W|} G_i^2}$

15) Para $i = 1$ até $|W|$ Faça $\lambda_i = max(0, \lambda_i + T.G_i)$ Fim-Para

16) Fim-Enquanto

17) Fim

Figura 3.2 - Algoritmo Subgradiente

No passo (1) definimos o valor inicial de π . Segundo Geoffrion[26], esse valor deve ser igual a 2 no início do algoritmo e se após j iterações o valor de $L(\lambda)$ não aumentar, fazemos $\pi = \frac{\pi}{2}$.

Nos passos (2) e (3), definimos respectivamente, os valores iniciais para os limites superior z_{UB} e inferior z_{LB} .

No passo (4) definimos o valor inicial do parâmetro $dif_{Z_{UB}Z_{LB}}$. Este valor representa a diferença percentual entre os valores z_{UB} (heurística lagrangeana) e z_{LB} (relaxação lagrangeana) que são atualizados em cada iteração do algoritmo do subgradiente.

No passo (5) definimos um valor para o *desvio* entre z_{UB} e z_{LB} . Este *desvio* será utilizado como limite inferior na comparação com valor $dif_{Z_{UB}Z_{LB}}$, atualizado em cada iteração do algoritmo.

No passo (6) definimos o total de iterações do algoritmo do subgradiente e no passo (7) definimos os valores iniciais dos multiplicadores λ_i . Esses multiplicadores serão atualizados ao longo das j iterações do algoritmo.

No passo (8) temos os critérios de parada do algoritmo associados respectivamente ao número de iterações do algoritmo e à diferença entre o limite superior e limite inferior.

No passo (10) resolvemos o problema relaxado utilizando os valores atuais de λ e obtemos uma solução x^* e um novo valor z_{LB} .

No passo (11) é executada a heurística lagrangeana para a obtenção uma solução viável para o PRR a partir da solução x^* e no passo (12) atualizamos o valor de $dif_{Z_{UB}Z_{LB}}$ a partir do cálculo da diferença percentual entre os valores z_{UB} e z_{LB} .

No passo (13) temos o cálculo dos subgradientes G_i associados às restrições relaxadas e no passo (14) temos a atualização do fator T . Esse passo depende do *gap* entre o valor atual do limite inferior z_{LB} e do limite superior z_{UB} . E no último passo, temos a atualização dos multiplicadores λ_i .

1) Faça $x^{**} = x^{*}$

2) Defina $W_{nc} = \{w_l \in W \mid d_{lk} > d, \forall\ v_k \in x^{**}\}$

3) Faça $tot_{desc} = |W_{nc}|$

4) Enquanto ($tot_{desc} > 0$) Faça

5) Selecione $w_l \in W_{nc}$

6) Determine $S_l = \{v_k \in V \backslash T \mid d_{lk} \leq d\}$ $(v_k \notin x^{**})$

7) Selecione aleatoriamente de S_l um vértice v_k

8) Adicione v_k entre dois vértices consecutivos v_i e v_j de x^{**} tal que $\{d_{ik} + d_{kj} - d_{ij}\}$ seja mínimo

9) Atualize W_{nc} (mesmo procedimento do passo(2))

10) Faça $tot_{desc} = |W_{nc}|$

11) Fim-Enquanto

12) Calcule z_H a partir de x^{**} construída pela heurística

13) Se ($z_{UB} > z_H$) Então Faça $z_{UB} = z_H$

Figura 3.3 - Heurística Lagrangeana

No passo (1) atribuímos a x^{**} a solução x^{*} proveniente da relaxação (rota não viável). Observamos que após aplicarmos a heurística, a solução x^{**} estará associada a uma rota viável para o PRR.

No passo (2) determinamos todos os vértices $w_l \in W$ que não são cobertos por x^{**} e no passo (3) associamos a tot_{desc} o total de vértices do conjunto W_{nc} .

No passo (4) temos a condição de parada da heurística. Ou seja, só teremos uma solução viável x^{**} , quando não houver nenhum vértice $w_l \in W$ descoberto.

No passo (5) selecionamos aleatoriamente um vértice w_l do conjunto W_{nc} , no passo (6) determinamos o conjunto S_l dos vértices v_k que não estão na rota e que cobrem o vértice w_l e no passo (7), selecionamos aleatoriamente um vértice v_k do conjunto S_l .

No passo (8) atualizamos a solução x^{**} inserindo o vértice v_k entre dois vértices consecutivos de x^{**} tal que a distância entre estes vértices e v_k seja mínima.

No passo (9) atualizamos o conjunto W_{nc} e no passo (10) atualizamos tot_{desc}.

No passo (12) calculamos o valor de z_H a partir da soma das distâncias entre os vértices v_i e v_j (consecutivos) que compõem a rota associada à solução viável x^{**}. E finalmente no passo (13), atualizamos o valor de z_{UB}, melhor limite superior obtido até o momento, caso o valor de z_H seja inferior ao valor de z_{UB}.

4 - Nova Regra de Redução para o PRR

4.1 - Introdução:

Conforme observado no capítulo 2, em alguns exemplos, ao aplicarmos a regra 1 e a regra 2 da literatura num grafo associado ao PRR geramos grafos reduzidos que apresentam em seu conjunto de soluções, rotas inviáveis para o grafo original. Isto ocorre, quando retiramos vértices do conjunto W sem a garantia de que existirá no grafo reduzido vértices do conjunto V que os cubram em qualquer rota viável.

Desta forma, para o desenvolvimento de uma nova regra de redução para o PRR, é necessário destacar quais vértices que devem realmente fazer parte do conjunto de soluções. Devemos retirar um vértice w_l do conjunto W se existir um vértice v_i do conjunto V que cubra w_l , ($d_{li} \leq d$) e que sempre tomará parte em qualquer rota viável associada ao PRR, ou seja, é necessário que o vértice v_i pertença a qualquer rota viável para garantir a cobertura do vértice w_l . A consideração dessa restrição evita o tipo de inviabilidade para o grafo original, decorrente da existência de vértices $w_l \in W$ não cobertos.

Com base nestas observações e pela necessidade de ter sempre todos os vértices do conjunto W cobertos por qualquer rota viável (condição do PRR), propomos a seguir, uma regra nova (RN) para o Problema de Recobrimento de Rotas.

4.2 - Descrição da Regra Nova:

Considere um vértice v_i do conjunto $V \backslash T$ como o único vértice que cobre um determinado vértice w_l do conjunto W , ou seja, $\delta_{li} = 1$ (δ_{li} assume valor 1 se $d_{li} \leq d$, e zero caso contrário). Levando em conta que o vértice $v_i \in V \backslash T$ é o único na cobertura do vértice $w_l \in W$, qualquer rota viável deverá passar por este vértice v_i e portanto, podemos transformá-lo em um vértice do conjunto T , $v_i \rightarrow t_i \in T$ (Brito[29]).

A racionalidade desta regra é baseada no seguinte fato: Se existe um único vértice $v_i \in V \backslash T$ cobrindo um vértice w_l do conjunto W , então esse vértice v_i deve obrigatoriamente tomar parte em qualquer rota viável, garantindo assim, que o vértice $w_l \in W$ será coberto.

Regra Nova (RN): Se existe um único vértice $v_i \in V \backslash T$ *tal que* $\delta_{li} = 1$, $w_l \in W$, *então o vértice* v_i *será tranformado em um vértice do conjunto* T $(v_i \rightarrow t_i \in T)$.

Com a aplicação da regra nova conjuntamente com as regras 3 e 4 descritas na seção 2.5, há a possibilidade de obter-se uma razoável redução no número de vértices dos conjuntos V e W associados ao grafo original.

Podemos observar, pela figura 4.1(a), que o vértice $v_i \in V\backslash T$ cobre o vértice $w_l \in W$, e é o único na cobertura deste vértice. Então, podemos aplicar na figura 4.1(a) a regra nova de redução, transformando o vértice $v_i \in V\backslash T$ em um vértice do conjunto T (figura 4.1(b)).

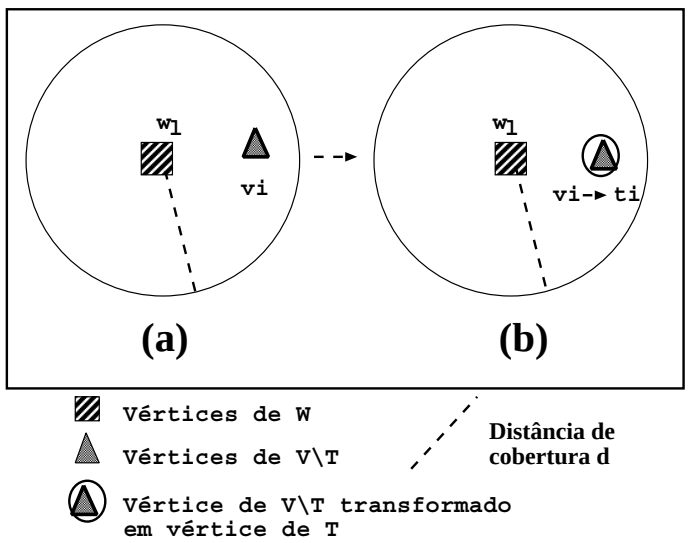

Figura 4.1 - Exemplo básico da aplicação da regra nova (RN) e da transformação do vértice $v_i \in V\backslash T$ no vértice $t_i \in T$

Na figura 4.2 apresentamos um novo exemplo da aplicação da regra nova, da regra 3 e da regra 4. Na figura 4.2(a) temos o resultado da aplicação da regra nova. Ao se aplicar a regra 3 no grafo da figura 4.2(a), os vértices w_l e w_m do conjunto W serão eliminados, pois agora estão cobertos por um vértice do conjunto T ($v_i \rightarrow t_i$), resultando no grafo reduzido apresentado na figura 4.2(b).

Em seguida, com a aplicação da regra 4 no grafo da figura 4.2(b), teremos a eliminação dos demais vértices do conjunto $V \backslash T$ que cobriam o vértice w_m do conjunto W , eliminado pela regra 3, resultando no grafo da figura 4.2(c). Conseqüentemente, após a aplicação destas três regras, teremos o grafo da figura 4.2(d) com apenas um vértice.

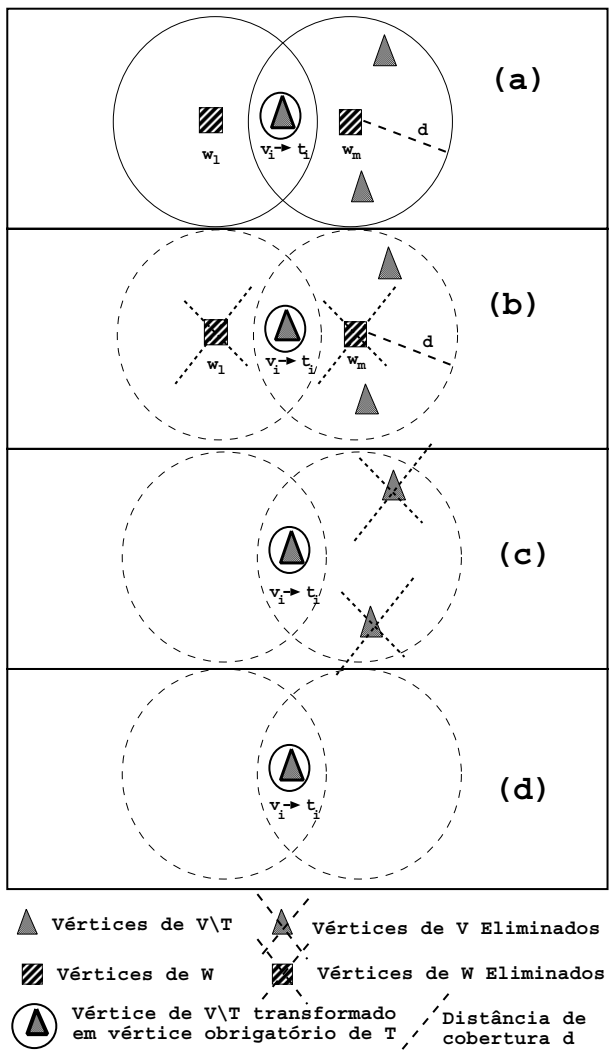

Figura 4.2 - Exemplo de grafo reduzido pela regra nova, regra 3 e regra 4

Na figura 4.3, damos um novo exemplo da aplicação da regra nova proposta neste capítulo, conjuntamente com as regras apresentadas na literatura (regras 3 e 4). Definimos um grafo original com 14 vértices, sendo $|V\backslash T| = 7$, $|T| = 1$ e $|W| = 6$.

Aplicando primeiramente a regra nova, transformamos 3 vértices do conjunto $V\backslash T$ em vértices obrigatórios do conjunto T . Portanto, agora temos $|V\backslash T| = 4$ e $|T| = 4$. Em seguida, aplicando a regra 3, eliminamos os vértices do conjunto W que são cobertos pelos vértices do conjunto T , e desta forma, eliminamos 6 vértices do conjunto W . Finalmente, ao aplicarmos a regra 4 eliminaremos 4 vértices do conjunto $V\backslash T$ que não cobrem nenhum vértice do conjunto W . O grafo reduzido é apresentado na figura 4.3(b).

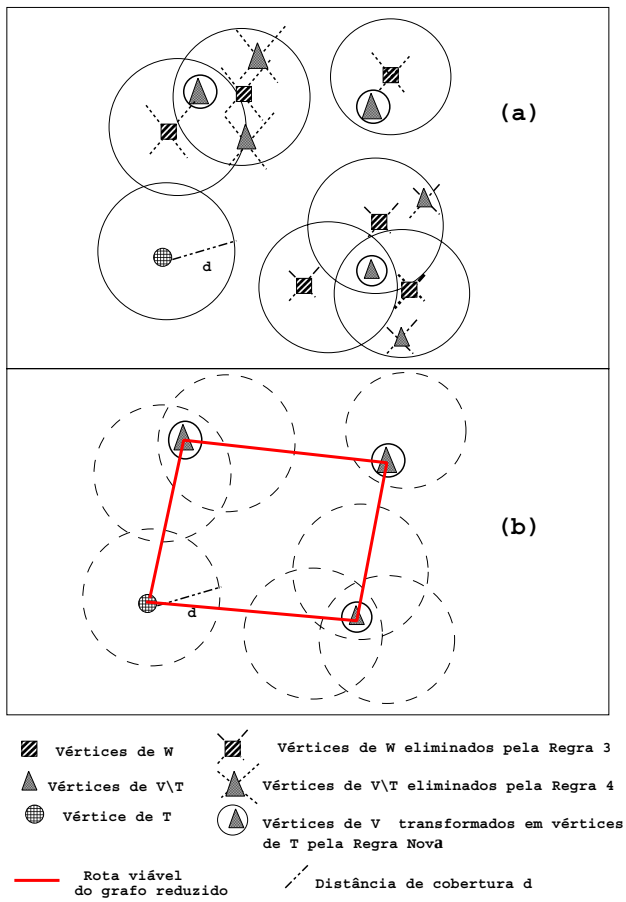

Figura 4.3 - Aplicação das regras nova, 3 e 4 em uma instância do PRR

Após a aplicação destas regras, obtemos um grafo reduzido com 4 vértices do conjunto T . Neste exemplo a redução foi de 71% e observamos que qualquer solução viável do grafo reduzido (figura 4.3(b)) será viável para o grafo original da figura 4.3(a).

5 - Algoritmos propostos para o PRR

5.1 - Introdução

Nas últimas décadas, a área de Otimização Combinatória tem recebido valiosas contribuições em termos de desenvolvimento de métodos exatos e aproximados, para resolução de diversos problemas.

Todavia, mesmo com todo este progresso, dependendo da dimensão dos problemas de otimização considerados, a ação e a eficácia dos métodos exatos pode ficar limitada. Ou seja, pode ser impossível encontrar uma solução ótima ou até mesmo viável em "tempo computacional admissível ".

O termo "tempo computacional admissível "está relacionado com o tipo de problema, sendo considerado assim quando não ultrapassar o limite de tempo de espera permitido ou aceitável (Viana[30]).

Uma possível saída encontrada para tentar contornar este tipo de problema é a de implementar heurísticas, de forma a obter soluções viáveis. Heurísticas são algoritmos que usam conhecimento acerca do problema que está sendo resolvido de modo a encontrar rapidamente e com alta probabilidade uma solução viável satisfatória.

Apesar das vantagens dos procedimentos heurísticos em termos de velocidade e qualidade de soluções, certos problemas tornam difícil o seu uso em algumas situações. Por exemplo, tem-se conhecimento de que a qualidade da solução de um algoritmo heurístico de otimização pode variar de acordo com as características da instância a ser resolvida. Uma heurística pode ser eficiente para algumas instâncias de um certo problema e por outro lado pode ter comportamento insatisfatório para outras instâncias do mesmo problema.

Outro problema que ocorre na utilização de heurísticas para problemas de otimização combinatória, está relacionado à tendência que elas possuem de ficarem presas em ótimos locais. As heurísticas que procuram sempre melhorar o valor da função objetivo, podem facilmente ficar estagnadas nestes pontos, sem a possibilidade de alcançar o ótimo global.

Como uma forma de generalizar o procedimento de busca heurística, foram propostas nas últimas décadas, uma série de heurísticas gerais, conhecidas como *Metaheurísticas*. Metaheurísticas são esquemas que podem ser implementados para diversos problemas. Uma metaheurística não especifica detalhes que podem ser adaptados ao problema em questão, mas indica apenas os elementos importantes que devem estar presentes em qualquer implementação.

Normalmente, uma metaheurística pode ser reutilizada em diversos domínios de problemas diferentes, variando-se apenas alguns dos seus parâmetros de funcionamento.

Outra característica importante das metaheurísticas é o uso de procedimentos aleatórios. Através do uso adequado destes procedimentos, a metaheurística reduz o risco de estagnação em ótimos locais durante o processo de busca. Deste modo, tenta-se eliminar o principal problema existente nas heurísticas de busca local. O uso de procedimentos aleatórios também favorece uma melhor exploração do espaço de soluções, permitindo que, em alguns casos, as soluções possam ser encontradas mais rapidamente.

Uma vantagem importante das metaheurísticas diz respeito a seu comportamento para grandes instâncias de problemas combinatórios. Devido às características apontadas, as metaheurísticas são capazes de encontrar soluções satisfatórias mesmo para instâncias grandes de problemas NP-Difíceis (Swarficter[2]), que não podem ser abordados com métodos exatos. Além disso, para alguns problemas onde boas heurísticas não foram ainda desenvolvidas, as metaheurísticas permanecem como uma alternativa razoável de solução.

A grande quantidade de problemas de otimização de difícil solução, encontrados em diferentes áreas, tem incentivado a pesquisa de diversas metaheurísticas eficientes, capazes de lidar com problemas cada vez mais complexos e dinâmicos.

Nos últimos anos, algumas metaheurísticas têm se sobressaído na solução de problemas de otimização combinatória pelos bons resultados computacionais obtidos. Dentre estas, podemos destacar, a Busca Tabu (Glover[31]), o VNS (*Variable Neighborhood Search*) (Hansen[32,33]) e o GRASP (*Greedy Randomized Adaptive Search Procedure*) (Resende[34]).

Devido ao bom desempenho obtido na aplicação destas metaheurísticas para muitos problemas de otimização combinatória e pela própria complexidade do Problema de Recobrimento de Rotas, desenvolvemos também, um conjunto de algoritmos baseados nas metaheurísticas GRASP e VNS.

Com a utilização destes algoritmos, há a possibilidade de se obter soluções viáveis, de "boa qualidade", para diversas instâncias do PRR. Ademais, é possível utilizar o valor associado a estas soluções como um limite superior para a formulação proposta para o PRR no capítulo 3.

Com a finalidade de fornecer um bom entendimento dos algoritmos propostos neste capítulo para a resolução do PRR, faremos inicialmente uma descrição concisa das metaheurísticas GRASP , VNS e das heurísticas Construtivas.

5.2 - Metaheurística GRASP

O GRASP, desenvolvido em 1995 por Tom Feo e Maurício Resende (Resende[34]), é um procedimento iterativo constituído por duas fases: Na primeira fase, conhecida como fase de construção, obtemos uma solução viável x_0 , cuja vizinhança será investigada durante a segunda fase (busca local), até que um ótimo local seja encontrado. A melhor de todas as soluções é dada como o resultado do GRASP (figura 5.1) após um número pré-determinado de n iterações.

Uma vizinhança $V(x_0)$ de uma solução x_0 é um conjunto de soluções que podem ser obtidas a partir de x_0 através da inserção, mudança de posição ou remoção de um elemento de x_0 .

Algoritmo GRASP Básico: Fase de Construção e de Busca Local (Minimização)
1) Defina $f^* = \infty$
2) Para $k = 1$ até n Faça
3) Aplique o procedimento de construção para obter uma solução viável x_0
4) Aplique a busca local, gerando uma nova solução $x_0' \in V(x_0)$
5) Se ($f(x_0') < f^*$) Então
6) Defina $x^* = x_0'$
7) Defina $f^* = f(x_0')$
8) Fim-Se
9) Fim-Para
10) Defina Solução-GRASP = x^*

Figura 5.1 - GRASP Básico

Na fase de construção, uma solução é iterativamente construída, elemento por elemento. Em cada iteração desta fase (figura 5.2) temos uma lista de candidatos (LC) que é formada por todos os elementos que podem ser incorporados na solução parcial x e que não provocam a inviabilidade do problema.

Uma vez definida a LC , avaliamos todos os seus elementos através de uma função gulosa $g(.)$, que representa o custo de se adicionar um novo elemento $t \in LC$ na solução parcial x .

A avaliação de todos os elementos de LC através desta função $g(.)$ implica na criação de uma lista de candidatos restrita (LCR) , formada pelos melhores elementos de LC , ou seja, aqueles que quando incorporados à solução parcial x produzem um acréscimo mínimo (no caso do problema de minimização). Este processo dá o aspecto guloso do GRASP, pois temos sempre os "melhores" elementos para serem incorporados na solução, na fase de construção.

O elemento $t \in LCR$ incorporado na solução parcial x é escolhido aleatoriamente. Esta forma de selecionar o elemento para compor a solução parcial x é que dá o aspecto probabilístico do GRASP. Feita a seleção de um elemento $t \in LCR$ para compor a solução parcial x , a LC é atualizada e os seus elementos são reavaliados utilizando a função $g(.)$. Este processo de reavaliação define o aspecto adaptativo do GRASP.

Algoritmo GRASP Básico: Fase de Construção	
1)	Defina $x = \{\ \}$
2)	Defina a lista de candidatos LC
3)	Avalie $g(.)$ considerando todos elementos $t \in LC$
4)	Enquanto $(LC \neq \emptyset)$ Faça
5)	Defina $s_i = min\{g(t) \mid t \in LC\}$
6)	Defina $s_s = max\{g(t) \mid t \in LC\}$
7)	Defina $LCR = \{t \in LC \mid g(t) \leq s_i + \alpha\ (s_s - s_i)\}$
8)	Selecione t de LCR aleatoriamente
9)	Defina $x = x \cup \{t\}$
10)	Defina $LC = LC \setminus \{t\}$ (Atualização)
11)	Reavalie $g(.)$ para todos $t \in LC$
12)	Fim-Enquanto
13)	Defina $x_0 = x$ (Solução Inicial)

Figura 5.2 - Algoritmo GRASP Básico - Fase de Construção

Para a construção da LCR , considerando um problema de minimização, a cada iteração na fase de construção, denotamos respectivamente por s_i e s_s o menor

e maior acréscimo provocados pela inserção de um elemento $t \in LC$ na solução, segundo a função gulosa $g(.)$.

O tamanho da LCR poderá ser limitado pelo número de elementos ou pelo parâmetro α que controla o nível de "gulosidade "e aleatoriedade do processo de construção. Pelo passo (7) (Figura 5.2) observamos que para $\alpha = 0$, o procedimento de construção torna-se guloso, pois a LCR tem apenas um elemento (o elemento que dá o menor acréscimo s_i), e quando $\alpha = 1$, todos os candidatos de LC estão em LCR produzindo desta forma uma solução aleatória.

Uma característica associada à fase de construção do GRASP é a de gerar soluções, que na maioria dos casos, possuem boa qualidade. Isto evita que o procedimento de busca local precise realizar um grande esforço computacional para melhorar a solução inicial x_0 e conseqüentemente, melhorar a eficiência da metaheurística.

Na segunda fase, fase de busca local, há uma tentativa de se melhorar a solução inicial x_0 obtida na primeira fase, tendo em vista que esta solução não é necessariamente uma solução ótima para o problema. A busca local consiste na substituição da solução x_0 pela melhor solução x_0' encontrada na vizinhança de x_0 , ou seja, procuramos então, $x_0' \in V(x_0)$ tal que $f(x_0') < f(x_0)$ (minimização).

Se não existir uma melhor solução nesta vizinhança, a solução corrente é considerada um ótimo local e a busca é concluída. A solução obtida a partir do GRASP, será a melhor das soluções obtidas em todas as iterações.

Os principais fatores responsáveis pelo sucesso da busca local são: A escolha eficiente da estrutura de vizinhanças, a eficiência da técnica de busca empregada nesta vizinhança e a solução inicial x_0 fornecida para esta busca.

Conforme descrito na literatura, a qualidade do resultado obtido a partir da aplicação do GRASP está diretamente ligada à variância das soluções geradas. A cada iteração do GRASP é produzida uma solução que pode ter uma distribuição diferente de todas as soluções obtidas até a iteração atual, sendo este fato proveniente da lista de candidatos restrita (LCR) . Se a lista de candidatos restrita contiver apenas um elemento, então apenas uma solução é produzida, acarretando em uma variância nula.

Por outro lado, se a cardinalidade da LCR for maior que 1, diferentes soluções podem ser produzidas, implicando em uma variância maior e conseqüentemente na obtenção de soluções melhores.

Existem vários procedimentos que podem ser inseridos no GRASP de forma que tenhamos soluções melhores do que aquelas obtidas simplesmente com a aplicação

do GRASP básico. Dentre estes, faremos uma descrição dos procedimentos de Filtro e de Reconexão por Caminhos (Path Relinking), implementados nos algoritmos GRASP propostos neste trabalho.

Além destes procedimentos, implementamos uma heurística de inserção que foi utilizada na fase de construção dos algoritmos GRASP.

- FILTRO - Conforme já apresentado, em cada iteração do GRASP básico temos a geração de uma solução viável x_0 na fase de construção, e depois aplicamos a busca local com a finalidade de obter uma solução $x'_0 \in V(x_0)$, tal que $f(x'_0) < f(x_0)$. No caso do GRASP com FILTRO, são geradas m soluções viáveis na fase de construção e dentre estas, seleciona-se a melhor solução, para então se aplicar o procedimento de busca local. Este procedimento é realizado em todas as iterações do GRASP.

- RECONEXÃO POR CAMINHOS - Foi introduzida por Glover [31] como uma maneira de integrar as estratégias de intensificação e de diversificação em Busca Tabu.

É uma técnica que gera novas soluções explorando a trajetória que conecta soluções de boa qualidade. Partindo de uma solução (chamada de solução base, S_B), um caminho no espaço de soluções que leve a uma outra solução (chamada de solução guia, S_G) é gerado e explorado em busca de soluções melhores. Este caminho no espaço de soluções é gerado através da seleção de movimentos que levem à solução guia, introduzindo assim, atributos pertencentes a esta solução.

Um "caminho" é então gerado de S_B para S_G produzindo uma seqüência de soluções intermediárias, S_I

$$S_B = S_I^0, S_I^1, S_I^2, ..., S_I^r, S_I^{r+1}, ..., S_I^n = S_G,$$

onde em cada iteração, obtemos S_I^{r+1} a partir de S_I^r através de "movimentos"que devem ser aplicados à solução base (S_B) para que ela se iguale à solução guia (S_G).

Finalmente, quando o "caminho " é completado, uma ou mais soluções intermediárias S_I são selecionadas . O objetivo é verificar se há alguma solução intermediária S_I , durante a "transformação "de S_B em S_G , que seja melhor do que as soluções obtidas até momento.

Em 1999, Laguna e Marti [35] propuseram uma maneira de integrar o procedimento de Reconexão por Caminhos ao GRASP. Neste trabalho, o conceito de reconexão (relinking) é reformulado, já que as soluções encontradas nas iterações do GRASP são independentes entre si e não estão necessariamente conectadas por uma série de movimentos, como no caso da Busca Tabu.

Eles optaram então, por um processo de reconexão que realiza movimentos necessários para transformar uma solução base S_B em uma solução guia S_G .

A Reconexão por Caminhos pode ser vista como uma técnica que faz com que em cada reconexão, a solução base S_B torne-se mais parecida com a solução guia S_G , mesmo que estas soluções sejam totalmente diferentes no início. Este procedimento pode gerar soluções intermediárias que tenham o custo menor do que as soluções base e guia.

Um refinamento do GRASP com a Reconexão por Caminhos foi desenvolvido por Aiex et al.[36]. Neste refinamento, um conjunto constituído por soluções de alta qualidade, denominado *Conjunto Elite* é formado com as soluções encontradas nas k primeiras iterações do GRASP.

A construção e atualização deste conjunto é definida da seguinte maneira: No início do algoritmo, o Conjunto Elite está vazio. Ao final de cada iteração da busca local, é efetuada a inserção da solução no Conjunto Elite se esta satisfazer determinados critérios.

Em seguida, uma das soluções pertencentes ao Conjunto Elite é escolhida aleatoriamente e a reconexão será aplicada tomando-se a solução encontrada ao final da busca local como solução base S_B e a solução escolhida do Conjunto Elite como solução guia S_G . A melhor solução encontrada no processo de reconexão também é inserida no Conjunto Elite, considerando as seguintes situações:

- Conjunto Elite Incompleto: Se o Conjunto Elite ainda não foi totalmente preenchido, isto é, se o número de soluções armazenadas é menor que um número máximo de soluções deste conjunto (parâmetro definido na fase de implementação), então uma solução deve fazer parte do Conjunto Elite ser for diferente de cada uma das soluções que já estão armazenadas.

- Conjunto Elite Completo: Para atualizar o conjunto considerando uma nova solução, esta solução deve possuir custo menor que a pior solução (solução de custo maior) do Conjunto Elite e ser suficientemente diferente de cada uma das soluções que já estão no Conjunto Elite.

Para este processo de reconexão são combinados diversos outros critérios para definir a solução base e a solução guia. No trabalho de Resende[37] temos a descrição de alguns destes critérios, gerando variações da própria técnica.

5.3 - Metaheurística VNS

Os algoritmos de busca local realizam uma seqüência de modificações em uma vizinhança de uma solução x_0 , tentando melhorar o valor da função objetivo até que um ótimo local seja encontrado.

O problema é que em muitos casos, o ótimo local está "distante "do ótimo global, e desta forma, é necessário analisar o valor da função objetivo em outras vizinhanças a fim de encontrar soluções melhores. Esta estratégia tem sido agregada em algumas metaheurísticas que possuem um mecanismo para escapar de ótimos locais.

De forma geral, uma vizinhança $V(x_0)$ de uma solução x_0 é um conjunto de soluções que podem ser obtidas a partir de x_0 através da inserção, mudança de posição ou remoção de um elemento de x_0 .

O enfoque explorado pela metaheurística VNS (*Variable Neighborhood Search*), proposta por Nenad Mladenović [32,33] está baseado em uma sistemática troca de estruturas de vizinhanças associadas a um algoritmo de busca local.

Diferentemente das outras metaheurísticas, o VNS não segue uma trajetória, mas explora incrementalmente vizinhanças distantes da vizinhança associada à solução x_0 na tentativa de obter soluções melhores, utilizando um conjunto finito de vizinhanças V_k $(k = 1, ..., k_{max})$ previamente selecionadas, sendo $V_k(x_0)$ o conjunto de soluções na $k_{ésima}$ vizinhança da solução x_0 .

No algoritmo VNS básico, selecionamos aleatoriamente uma solução $x'_0 \in V_k(x_0)$ e submetemos esta solução a uma busca local. Se a busca local fracassar na obtenção de uma solução de valor melhor do que a solução x_0 , incrementa-se a ordem da vizinhança corrente, e desta forma, passamos a explorar a solução em outra vizinhança V_{k+1} .

Em caso contrário, se a busca local encontrar uma solução de valor melhor do que a solução x_0 na vizinhança V_k , atualizamos a solução e a ordem da vizinhança volta a ser um (retornamos à vizinhança V_1). A seguir, apresentamos o algoritmo VNS básico.

Algoritmo VNS - Variable Neighborhood Search

1) Defina k_{max} (Número máximo de estruturas de vizinhança)

2) Selecione o conjunto de estruturas de vizinhança V_k $(k = 1, ..., k_{max})$

3) Encontre uma solução inicial x_0

4) Defina um critério de parada

5) Defina $f^* = f(x_0)$

6) Enquanto (critério de parada não for satisfeito) Faça

7) Defina $k = 1$

8) Enquanto $(k < k_{max})$ Faça

9) Gere aleatoriamente uma solução x' tq $x' \in V_k(x_0)$

10) Aplique um procedimento de busca local em x' para obter x'' (ótimo local)

11) Se $(f(x'') < f^*)$ Então

12) Defina $x_0 = x''$

13) Defina $k = 1$

14) Defina $f^* = f(x_0)$

15) Senão

16) Defina $k = k + 1$

17) Fim-Se

18) Fim-Enquanto

19) Fim-Enquanto

Figura 5.3 - Algoritmo VNS - Variable Neighborhood Search

5.4 - Algoritmos Heurísticos Construtivos

Grande parte dos algoritmos heurísticos construtivos desenvolvidos para o PCV utilizam a estratégia de solução denominada *gulosa* (Goldbarg, Luna [38]). Tal estratégia é baseada na idéia de obter o maior proveito ou ganho possível, em cada passo da heurística. Naturalmente, a política global de tomada de decisão pode não comportar sempre ações que levem aos maiores ganhos imediatos.

Algumas heurísticas de construção possuem um processo de decisão mais elaborado, quando comparadas com as heurísticas simplesmente gulosas. Nesse processo de decisão, são envolvidos três níveis:

- A decisão de um ciclo inicial.
- A escolha do vértice a ser inserido na solução.
- A posição de inserção desse novo vértice.

Tais heurísticas são classificadas como heurísticas de inserção. Normalmente, tais heurísticas partem de uma rota parcial inicial (um ciclo normalmente de comprimento 3) e vão sistematicamente selecionando e inserindo vértices ainda não incluídos na solução até completar um ciclo (rota viável). Alguns dos critérios mais utilizados para a seleção dos vértices que não fazem parte da rota parcial inicial são:

- Inserção do vértice mais próximo.
- Inserção do vértice mais distante.
- Inserção mais barata (inserção do vértice que conduz ao ciclo mais barato).
- A inserção aleatória.

Considerando as inserções com os vértices mais próximos ou mais distantes de qualquer vértice da rota parcial inicial, e escolhido um vértice v_k , cabe decidir como este vértice será inserido na rota parcial inicial. Como o vértice ainda não pertence a rota parcial inicial, serão necessárias duas arestas para incluí-lo e a remoção da aresta (i, j) .

O critério de decisão sobre quais arestas devemos empregar nesse processo é expresso pelo efeito do balanço entrada *versus* saída de arestas (Goldbarg, Luna[38]). Considerando uma inserção entre os vértices v_i e v_j , o critério para a escolha do ponto de inserção se confunde com o critério de escolha da aresta (i, j) que minimize o seguinte balanço:

$$\{d_{i\,k} + d_{k\,j} - d_{i\,j}\}$$

No caso da inserção mais barata, o próprio vértice v_k passa a ser escolhido pelo critério de mínimo balanço, considerando-se todas as inserções possíveis para todos os vértices ainda não pertencentes à rota parcial inicial.

De uma forma geral, as heurísticas de inserção podem ser descritas pelos seguintes passos:

Heurística de Inserção	
1)	Iniciar por um ciclo de vértices $\{v_1\ v_2\ v_3\}$
2)	Encontrar o vértice v_k não pertencente ao ciclo, mais próximo
	ou mais distante de qualquer um dos vértices pertencentes ao ciclo
3)	Encontrar a aresta $(i\ ,j)$ tal que $\{d_{i\,k} + d_{k\,j} - d_{i\,j}\}$ seja mínimo
4)	Inserir o vértice v_k entre os vértices v_i e v_j
5)	Voltar ao passo (2) até que a rota se torne viável

Figura 5.4 - Heurística de Inserção

5.5 - Descrição dos Algoritmos propostos para o PRR

Com finalidade de gerar soluções viáveis para o PRR de uma forma razoavelmente rápida (quando comparado a um método exato) e utilizar o valor destas soluções como limite superior para o valor ótimo, desenvolvemos seis algoritmos baseados nas metaheurísticas GRASP e VNS.

Considerando que o GRASP utiliza um algoritmo na fase de construção e um algoritmo na fase de busca local, implementamos e combinamos os seguintes algoritmos: Vizinho Mais Próximo, VNS, Troca Simultânea, Filtro e Reconexão por Caminhos. A combinação destes algoritmos proporcionou a obtenção de seis algoritmos GRASP (tabela 5.1) denominados: GR1, GR2, GR1F, GR2F, GR1FPR-V0 e GR1FPR-V1.

Algoritmos GRASP	Fase de Construção	Fase Busca Local
GR1	Vizinho mais Próximo (Algoritmo 1)	VNS (Algoritmo 2)
GR2	Vizinho mais Próximo (Algoritmo 3)	Troca Simultânea (Algoritmo 4)
GR1F	Vizinho mais Próximo (Algoritmo 1) + Filtro (Algoritmo 5)	VNS (Algoritmo 2)
GR2F	Vizinho mais Próximo (Algoritmo 3) + Filtro (Algoritmo 5)	Troca Simultânea (Algoritmo 4)
GR1FPR-V0	Vizinho mais Próximo (Algoritmo 1) + Filtro (Algoritmo 5) + Reconexão por Caminhos (Algoritmo 6)	VNS (Algoritmo 2)
GR1FPR-V1	Vizinho mais Próximo (Algoritmo 1) + Filtro (Algoritmo 5)	Reconexão por Caminhos (Algoritmo 6) + VNS (Algoritmo 2)

Tabela 5.1 - Resumo dos algoritmos baseados nas Metaheurísticas GRASP e VNS

5.5.1 - Algoritmo 1 - Vizinho mais Próximo 1

O algoritmo 1 constrói uma $Rota_{Inicial}$ a partir de 2 vértices selecionados aleatoriamente do conjunto T, e para a inserção dos vértices restantes nesta rota (vértices de V), aplicamos a heurística de inserção do vértice mais próximo (Goldbarg, Luna [38]). Nesta heurística escolhemos aleatoriamente um vértice v_k da Lista de Candidatos Restrita (LCR) e inserimos este vértice entre dois vértices consecutivos da $Rota_{Inicial}$ (v_i e v_j) de forma que o acréscimo no custo da rota seja o menor possível. Deve-se encontrar o menor valor de $\{d_{ik}+d_{kj}-d_{ij}\}$, $\forall\, v_i, v_j$ consecutivos na $Rota_{Inicial}$.

Algoritmo 1: Vizinho mais próximo 1

1) Defina $Rota_{Inicial} = \{t_i - t_j - t_i\}$, $\{t_i, t_j\} \in T$, (selecionados aleatoriamente)

2) Defina $W_{nc} = \{w_l \in W \mid d_{lk} > d$, $\forall\, t_k \in Rota_{Inicial}\}$

3) Defina $S_l = \{v_k \in V \setminus T \mid d_{lk} \leq d\}$, $\forall\, w_l \in W_{nc}$

4) Defina $LC = \{T \setminus \{t_i, t_j\}\} \cup S_l$

5) Enquanto $((W_{nc} \neq \emptyset)$ ou (LC tiver vértices de T)) Faça

6) Insira na LCR os p vértices da LC mais próximos de um vértice v_s da $Rota_{Inicial}$, escolhido aleatoriamente $(v_s \in V)$, $p = min(|LC|, 3)$

7) Selecione aleatoriamente da LCR um vértice v_k

8) Insira v_k entre 2 vértices consecutivos, v_i e $v_j \in Rota_{Inicial}$ de forma que o valor de $\{d_{ik} + d_{kj} - d_{ij}\}$ seja mínimo

9) Defina $Rota_{Inicial} = Rota_{Inicial} + \{v_k\}$

10) Defina $J_k = \{w_l \in W_{nc} \mid d_{lk} \leq d\}$ (v_k vértice inserido no passo (9))

11) Defina $W_{nc} = W_{nc} \setminus J_k$

12) Defina $LC = LC \setminus \{v_k\}$

13) Fim-Enquanto

14) Defina $Custo_{Inicial} = \sum d_{ij}$ $(\forall\, v_i, v_j \in Rota_{Inicial})$ $(v_i, v_j$ vértices consecutivos da rota)

Figura 5.5 - Algoritmo 1: Vizinho mais próximo

No passo (1), iniciamos a construção da $Rota_{Inicial}$ com 2 vértices selecionados aleatoriamente do conjunto T. No passo (2) definimos W_{nc} como o conjunto dos vértices de W ainda não cobertos pela $Rota_{Inicial}$. No passo (3), para cada vértice $w_l \in W_{nc}$, temos definido um conjunto S_l, que contém todos os vértices $v_k \in V \backslash T$ que cobrem o vértice w_l.

Em seguida, no passo (4), construímos uma lista de candidatos (LC), que contém todos os vértices de T (excluindo t_i e t_j inseridos na $Rota_{Inicial}$) e os vértices v_k de cada conjunto S_l. É a partir desta lista que iremos escolher os vértices que serão inseridos na $Rota_{Inicial}$, utilizando o algoritmo de inserção do vértice mais próximo. No passo (5) temos o critério de parada do algoritmo.

No passo (6) é definida a lista de candidatos restrita (LCR), considerando os p vértices ($p = min(|LC|, 3)$) da LC mais próximos do vértice v_s escolhido aleatoriamente da $Rota_{Inicial}$. A LCR poderá ser limitada pelo número de elementos em seu conjunto ou pelo parâmetro α que controla a aleatoriedade do algoritmo (vide seção 5.2).

No passo (7) selecionamos aleatoriamente da LCR um vértice v_k que será inserido entre dois vértices consecutivos v_i e v_j da $Rota_{Inicial}$, tal que a distância deste vértice a rota seja mínima (passo (8)). No passo (9) atualizamos a $Rota_{Inicial}$ com este novo vértice.

Com a inserção do vértice $v_k \in V$ na $Rota_{Inicial}$, determinamos o conjunto J_k, que é o conjunto dos vértices $w_l \in W_{nc}$ cobertos pelo vértice v_k (passo (10)). No passo (11) retiramos os vértices $w_l \in J_k$ do conjunto W_{nc}, tendo em vista que estes vértices são cobertos pelo vértice v_k que foi inserido na $Rota_{Inicial}$. E no passo(12), atualizamos a lista de candidatos, retirando este vértice.

No passo (14), definimos o $Custo_{Inicial}$ da rota como a soma das distâncias entre vértices consecutivos v_i e $v_j \in Rota_{Inicial}$.

Observamos que a variabilidade das soluções geradas em cada iteração deste algoritmo, é decorrente da escolha aleatória de dois vértices obrigatórios iniciais $t_i, t_j \in T$ (passo 1), da formação da lista de candidatos restrita (passo 6) e da escolha do vértice v_k ser aleatória (passo 7).

5.5.2 - Algoritmo 2 - VNS

O algoritmo 2 efetua uma busca local com a finalidade de melhorar o custo da $Rota_{Inicial}$ (solução inicial viável) obtida no algoritmo 1. Neste algoritmo utilizamos uma estrutura de vizinhanças V_α , $(\alpha = 1, .., \alpha_{max})$, baseada na troca simultânea de α vértices v_k do conjunto $V \backslash T$ pertencentes à $Rota_{Inicial}$, com outros α vértices quaisquer do conjunto V , diferentes entre si e que cobrem os mesmos vértices w_l que v_k cobre.

Seja α_{max} o total de vértices $v_k \in V \backslash T$ que estão na $Rota_{Inicial}$ e que serão substituídos por outros vértices do conjunto V . Neste procedimento, V_1 indica que trocaremos 1 vértice $v_k \in V \backslash T$ da $Rota_{Inicial}$ por outro vértice v_i do conjunto V sendo $v_k \neq v_i$; V_2 indica que trocaremos dois vértices v_k e $v_j \in V \backslash T$ da $Rota_{Inicial}$ por outros dois vértices do conjunto V diferentes entre si. Analogamente, $V_{\alpha_{max}}$ indica que trocaremos α_{max} vértices do conjunto $V \backslash T$ na $Rota_{Inicial}$ por outros α_{max} vértices do conjunto V diferentes entre si.

No passo (1) do algoritmo 2, definimos $Rota_{Melhor}$, $Rota_{Nova}$, $Custo_{Melhor}$ e $Custo_{Novo}$ com as informações da $Rota_{Inicial}$ e do $Custo_{Inicial}$, oriundos do algoritmo 1. Em seguida, no passo (3), executamos o procedimento I caso a $Rota_{Inicial}$ seja formada por apenas vértices do conjunto T . Neste procedimento é efetuada a troca de posição entre os vértices $t \in T$, na tentativa de reduzir o custo da $Rota_{Inicial}$.

Da mesma forma, se houver pelo menos um vértice $v_k \in V \setminus T$ na $Rota_{Inicial}$, há a possibilidade de ocorrer uma melhora no custo desta rota através da troca deste(s) vértice(s) v_k por vértice(s) v_i do conjunto V . Inicialmente, nos passos (4) e (5), definimos os limitantes da vizinhança, ou seja, $\alpha = 1$ e α_{max} é igual ao total de vértices v_k na $Rota_{Melhor}$ $(v_k \in V \backslash T)$. No passo (6) definimos a estrutura de vizinhança V_α , com base no número total de vértices do conjunto $V \backslash T$ pertencentes a $Rota_{Melhor}$.

Enquanto não for atingida a última estrutura de vizinhança α_{max} (passo (7)), escolhemos no passo (8) α vértices $v_k \in V \backslash T$ pertencentes a $Rota_{Melhor}$. Para cada vértice v_k escolhido (passo (9)), definimos no passo (10) o conjunto J_k , que é o conjunto de vértices $w_l \in W$ que são cobertos pelo vértice v_k . No passo (11), determinamos para cada $w_l \in J_k$ outros vértices $v_i \in V$ $(v_i \neq v_k)$. No passo (12) definimos o conjunto S a partir da união dos conjuntos S_l .

No passo (14) selecionamos aleatoriamente do conjunto S um vértice v_i para ser inserido no lugar de $v_k \in Rota_{Melhor}$ e no passo (15) atualizamos a $Rota_{Melhor}$, inserindo no lugar do vértice v_k o vértice v_i . No passo (18) executamos o procedimento II.

Ainda neste procedimento, definimos o $Custo_{Melhor}$ considerando a soma das distâncias entre os vértices consecutivos da $Rota_{Melhor}$ e verificamos se $Custo_{Melhor}$ é menor que o $Custo_{Novo}$. Caso isto ocorra, definimos o $Custo_{Novo} = Custo_{Melhor}$ e a $Rota_{Nova} = Rota_{Melhor}$ como a melhor solução encontrada e retornamos para a vizinhança V_1 utilizando a $Rota_{Melhor}$. Se não ocorrer uma redução no custo, mudamos de vizinhança e trocamos $\alpha + 1$ vértices v_k na iteração seguinte.

Algoritmo 2 : VNS

1) Defina $Rota_{Melhor} = Rota_{Inicial}$, $Custo_{Melhor} = Custo_{Inicial}$
e Defina $Rota_{Nova} = Rota_{Inicial}$, $Custo_{Novo} = Custo_{Inicial}$ (Algoritmo 1)

2) Se ($Rota_{Inicial}$ contém somente vértices de T) e ($|T| > 3$)

3) Então Execute o Procedimento I (página 55)

4) Senão Defina $\alpha = 1$

5) Defina α_{max} = Total de vértices v_k na $Rota_{Melhor}$ ($v_k \in V \backslash T$)

6) Defina a estrutura de vizinhança V_α ($\alpha = 1, .., \alpha_{max}$)

7) Enquanto ($\alpha \leq \alpha_{max}$) Faça

8) Escolha aleatoriamente α vértices v_k da $Rota_{Melhor}$

9) Para cada v_k escolhido Faça

10) Defina $J_k = \{\forall\ w_l \in W \mid d_{lk} \leq d\}$

11) Defina $S_l = \{v_i \in V \mid d_{li} \leq d\}$ ($\forall\ w_l \in J_k$, $v_i \neq v_k$)

12) Defina $S = \cup S_l$

13) Se ($S \neq \emptyset$) Então

14) Escolha aleatoriamente de S um vértice v_i

15) Defina $Rota_{Melhor} = Rota_{Melhor} \backslash \{ v_k\} + \{v_i\}$

16) Fim-Se

17) Fim-Para

18) Execute Procedimento II

19) Fim-Enquanto

20) Fim-se

Figura 5.6 - Algoritmo 2 : VNS

Procedimento I:

3.1) Para $k = 1$ até $|T|$ Faça

3.2) Troque o vértice t_i na k-ésima posição por outro vértice t_j escolhido aleatoriamente na p-ésima posição $(p \neq k)$ $(t_i, t_j \in Rota_{Melhor})$

3.3) Defina $Custo_{Melhor} = \sum d_{ij}$ $(\forall\, t_i, t_j \in Rota_{Melhor})$ (vértices consecutivos)

3.4) Se $(Custo_{Melhor} < Custo_{Novo})$ Então

3.5) Defina $Custo_{Novo} = Custo_{Melhor}$

3.6) Defina $Rota_{Nova} = Rota_{Melhor}$

3.7) Senão

3.8) Defina $Rota_{Melhor} = Rota_{Nova}$

3.9) Fim-Se

3.10) Fim-Para

Figura 5.7 - Procedimento I

Procedimento II:

18.1) Se ($Rota_{Melhor}$ é Viável)

18.2) Então Defina $Custo_{Melhor} = \sum d_{ij}$ $(\forall\, v_i, v_j \in Rota_{Melhor})$

18.3) Se ($Custo_{Melhor} < Custo_{Novo}$)

18.4) Então Defina $Custo_{Novo} = Custo_{Melhor}$

18.5) Defina $Rota_{Nova} = Rota_{Melhor}$

18.6) Defina $\alpha = 1$

18.7) Senão Defina $\alpha = \alpha + 1$

18.8) Defina $Rota_{Melhor} = Rota_{Nova}$

18.9) Fim-Se

18.10) Senão Defina $Rota_{Melhor} = Rota_{Nova}$

18.11) Fim-Se

Figura 5.8 - Procedimento II

Com a combinação dos algoritmos 1 e 2, obtemos o algoritmo GR1:

Algoritmo GR1 : Vizinho mais Próximo + VNS
1) Defina $Custo_{final} = \infty$
2) Para $k = 1$ até 1000 Faça
3) Aplique o algoritmo de construção (Algoritmo 1) para obter ($Rota_{Inicial}$, $Custo_{Inicial}$)
4) Aplique a busca local (Algoritmo 2) considerando a ($Rota_{Inicial}, Custo_{Inicial}$) e gere uma nova solução ($Rota_{Melhor}$, $Custo_{Melhor}$)
5) Se ($Custo_{Melhor} < Custo_{Final}$) Então
6) Defina $Custo_{Final} = Custo_{Melhor}$
7) Defina $Rota_{Final} = Rota_{Melhor}$
8) Fim-se
9) Fim-para
10) Defina Solução-GRASP $= Rota_{Final}$

Figura 5.9 - Algoritmo GR1

No passo(1) inicializamos o $Custo_{Final}$ e no passo (2) temos o critério de parada do algoritmo GR1, baseado no número máximo de iterações. No passo (3), que se refere a fase de construção, aplicamos o algoritmo 1 e obtemos a $Rota_{Inicial}$. No passo (4) aplicamos o algoritmo 2 e obtemos a $Rota_{Melhor}$. Se $Custo_{Melhor}$ for menor do que o $Custo_{Final}$ (passo(5)) então, nos passos (6) e (7) definimos $Custo_{Final} = Custo_{Melhor}$ e $Rota_{Final} = Rota_{Melhor}$

5.5.3 - Algoritmo 3 - Vizinho mais Próximo 2

Os algoritmos 1 e 3 diferem basicamente pela ordem de inserção dos vértices na $Rota_{Inicial}$. No algoritmo 3, temos a construção da $Rota_{Inicial}$ inserindo primeiramente todos os vértices do conjunto T (Procedimento III). Em seguida, verificamos se há algum vértice w_l do conjunto W que não esteja coberto por pelo menos um dos vértices t_k pertencentes a $Rota_{Inicial}$. Se isto ocorrer, faremos a inserção de alguns vértices do conjunto $V \backslash T$ na $Rota_{Inicial}$ com a finalidade cobrir todos os vértices w_l (Procedimento IV).

Algoritmo 3: Vizinho mais próximo 2
1) Defina $Rota_{Inicial} = \{t_i - t_j - t_i\}$, $\{t_i, t_j\} \in T$ (selecionados aleatoriamente)
2) Defina $W_{nc} = \{ w_l \in W \mid d_{lk} > d \}$ ($\forall\, t_k \in Rota_{Inicial}$)
3) Defina $LC = T \backslash \{ t_i, t_j \}$
4) Execute o Procedimento III
5) Se $(W_{nc} \neq \emptyset)$ Então Execute o Procedimento IV
6) Defina $Custo_{Inicial} = \sum d_{ij}$ $(\forall v_i, v_j \in Rota_{Inicial}$ (v_i, v_j vértices consecutivos))

Figura 5.10 - Algoritmo 3: Vizinho mais próximo

Procedimento III:
4.1) Enquanto $(LC \neq \emptyset)$ Faça
4.2) Insira na LCR os p vértices da LC mais próximos de um vértice t_s da $Rota_{Inicial}$, escolhido aleatoriamente $(p = min(\|LC\|, 3))$.
4.3) Selecione aleatoriamente da LCR um vértice t_k .
4.4) Insira t_k entre 2 vértices consecutivos t_i e $t_j \in Rota_{Inicial}$ de forma que o valor de $\{d_{ik} + d_{kj} - d_{ij}\}$ seja mínimo
4.5) Defina $Rota_{Inicial} = Rota_{Inicial} + \{ t_k \}$
4.6) Defina $J_k = \{w_l \in W_{nc} \mid d_{lk} \leq d\}$ (t_k vértice inserido no passo (4.5))
4.7) Defina $W_{nc} = W_{nc} \setminus J_k$ e Defina $LC = LC \setminus \{t_k\}$
4.8) Fim-Enquanto

Figura 5.11 - Procedimento III

Procedimento IV:
5.1) Enquanto $(W_{nc} \neq \emptyset)$ Faça
5.2) Defina $S_l = \{v_k \in V \backslash T \mid d_{lk} \leq d\}$, $\forall\, w_l \in W_{nc}$
5.3) Defina $LC = \cup\, S_l$
5.4) Insira na LCR os p vértices da LC mais próximos de um vértice v_s da $Rota_{Inicial}$ escolhido aleatoriamente $(p = min(\|LC\|, 3))$
5.5) Selecione aleatoriamente da LCR um vértice v_k
5.6) Insira v_k entre 2 vértices consecutivos v_i e $v_j \in Rota_{Inicial}$ de forma que o valor de $\{d_{ik} + d_{kj} - d_{ij}\}$ seja mínimo
5.7) Defina $Rota_{Inicial} = Rota_{Inicial} + \{v_k\}$
5.8) Defina $J_k = \{w_l \in W_{nc} \mid d_{lk} \leq d\}$ (v_k vértice inserido no passo (5.7))
5.9) Defina $W_{nc} = W_{nc} \setminus J_k$
5.10) Fim-Enquanto

Figura 5.12 - Procedimento IV

No passo (1) iniciamos a construção da $Rota_{Inicial}$ com 2 vértices selecionados aleatoriamente do conjunto T . No passo (2) definimos W_{nc} como o conjunto de vértices do conjunto W ainda não cobertos pela $Rota_{Inicial}$. Em (3) definimos uma lista de candidatos (LC) que contém todos os vértices do conjunto T , excluindo t_i e t_j inicialmente inseridos na $Rota_{Inicial}$.

No passo (4) executamos o Procedimento III. Enquanto houver um vértice do conjunto T na LC , temos a inserção na LCR dos p vértices da LC mais próximos de um vértice t_s selecionado aleatoriamente da $Rota_{Inicial}$ (passo 4.2).

No passo (4.3) selecionamos aleatoriamente da LCR um vértice t_k que será inserido entre dois vértices consecutivos t_i e $t_j \in Rota_{Inicial}$, de forma que a distância deste vértice à $Rota_{Inicial}$ seja mínima (passo (4.4)). No passo (4.5) atualizamos a $Rota_{Inicial}$ com este novo vértice.

Com a inserção do vértice $t_k \in T$ na $Rota_{Inicial}$, determinamos J_k , que é o conjunto de vértices $w_l \in W_{nc}$ cobertos pelo vértice t_k (passo (4.6)).

No passo (4.7), retiramos do conjunto W_{nc} os vértices que estão em J_k, tendo em vista que estes vértices agora são cobertos pelo vértice t_k inserido na $Rota_{Inicial}$. E atualizamos a LC retirando o vértice t_k.

Em seguida, após a execução do procedimento III, verificamos (passo (5)) se $W_{nc} \neq \emptyset$. Caso isto ocorra, efetuamos inserção de alguns vértices do conjunto $V \backslash T$ na $Rota_{Inicial}$ com a finalidade de cobrir todos os vértices do conjunto W_{nc}.

Com a execução do Procedimento IV (passo (5)), definimos para cada $w_l \in W_{nc}$ um conjunto S_l que contém todos os vértices $v_k \in V \backslash T$ que cobrem o vértice w_l (passo (5.2)). Em seguida, no passo (5.3), construímos uma lista de candidatos LC que é formada pela união dos conjuntos S_l. No passo (5.4) inserimos na LCR os p vértices da LC mais próximos do vértice v_s selecionado aleatoriamente da $Rota_{Inicial}$ ($p = min(|LC|, 3)$).

No passo (5.5), selecionamos aleatoriamente da LCR um vértice v_k que será inserido entre dois vértices consecutivos v_i e $v_j \in Rota_{Inicial}$, de forma que a distância deste vértice a $Rota_{Inicial}$ seja mínima (passo (5.6)). E no passo (5.7) inserimos na $Rota_{Inicial}$ este novo vértice.

No passo (5.8) determinamos o conjunto J_k, que é o conjunto dos vértices $w_l \in W_{nc}$ cobertos pelo vértice v_k e no passo (5.9) atualizamos o conjunto W_{nc}.

Finalmente, após a execução do procedimento IV, efetuamos no passo (6) o cálculo do custo da $Rota_{Inicial}$.

5.5.4 - Algoritmo 4 - Troca simultânea

Este algoritmo de busca local utiliza uma rota inicial viável como ponto de partida, e tenta através da troca simultânea dos vértices do conjunto $V \setminus T$, melhorar o custo desta rota.

O algoritmo 4 efetua uma busca local trocando os vértices $v_k \in V \setminus T$ que estão na $Rota_{Inicial}$ com outros vértices $v_i \neq v_k$ do conjunto $V \setminus T$, que cobrem os mesmos vértices w_l que v_k cobre, com a finalidade de reduzir o custo da rota. Neste processo, ao se trocar os vértices do conjunto $V \setminus T$ entre si, deve-se preservar a cobertura de todos os vértices do conjunto W.

No algoritmo 4 trocamos em cada iteração, um vértice do conjunto $V \setminus T$ que está na $Rota_{Inicial}$ de acordo com a ordem em que este vértice aparece na rota.

No passo (1) definimos $Rota_{Melhor}$ e o $Custo_{Melhor}$ a partir das informações do algoritmo 3. No passo (3), executamos o procedimento I (descrito na página 55) caso a $Rota_{Inicial}$ seja formada apenas por vértices do conjunto T.

Caso haja pelo menos um vértice $v_k \in V \setminus T$ na $Rota_{Inicial}$, há a possibilidade de reduzir o custo desta rota através da troca deste vértice v_k por um outro vértice v_i do conjunto $V \setminus T$. Desta forma, no passo (4), inicializamos $n = 1$ e no passo (5) definimos n_{max} a partir do total de vértices $v_k \in V \setminus T$ que estão na $Rota_{Inicial}$.

No passo (6) temos o critério de parada no algoritmo, associado ao número total de vértices v_k que deverão ser trocados na $Rota_{Inicial}$. No passo (7) escolhemos o n-ésimo vértice v_k da $Rota_{Inicial}$ e no passo (8) definimos o conjunto J_k, observando quais são os vértices $w_l \in W$ que o vértice v_k cobre. Em seguida, no passo (9), definimos para cada $w_l \in J_k$, quais são os outros vértices v_i que cobrem w_l. No passo (10) definimos o conjunto S a partir da união dos conjuntos S_l.

Se o conjunto S for não vazio (passo (11)), será efetuada a troca de v_k por cada vértice $v_j \in S$ (passo (13)). No passo (14) definimos a $Rota_{Nova}$ a partir da $Rota_{Inicial}$ e da troca do vértice v_k pelo vértice v_j. No passo (15) definimos o $Custo_{Novo}$ relacionado com a $Rota_{Nova}$ e no passo (16), executamos o Procedimento V para verificar a viabilidade e atualizar o custo da $Rota_{Melhor}$.

Algoritmo 4 : Troca Simultânea

1) Defina $Rota_{Melhor} = Rota_{Inicial}$ e $Custo_{Melhor} = Custo_{Inicial}$ (Algoritmo 3)

2) Se ($Rota_{Nova}$ contém apenas vértices do conjunto T)

3) Então Execute o Procedimento I (página 55).

4) Senão Defina $n = 1$

5) Defina n_{max} = Total de vértices $v_k \in V \backslash T$ na $Rota_{Inicial}$

6) Enquanto $(n \leq n_{max})$ Faça

7) Selecione o n-ésimo vértice $v_k \in V \backslash T$ da $Rota_{Inicial}$

8) Defina $J_k = \{w_l \in W \mid d_{lk} \leq d\}$

9) Defina $S_l = \{ v_i \in V \backslash T \mid d_{li} \leq d\}$ $(v_i \neq v_k, \forall w_l \in J_k)$

10) Defina $S = \cup S_l$

11) Se $(S \neq \emptyset)$ Então

12) Para $j = 1$ até $\left|S\right|$ Faça

13) Troque v_k por $v_j \in S$ (na j-ésima posição de S)

14) Defina $Rota_{Nova} = Rota_{Inicial} \setminus \{v_k\} + \{v_j\}$

15) Defina $Custo_{Novo} = \sum d_{ij}$ $(\forall\, v_i, v_j \in Rota_{Nova}$)
$(v_i, v_j$ vértices consecutivos da rota)

16) Execute o Procedimento V

17) Fim-Para

18) Fim-Se

19) Defina $n = n + 1$

20) Fim-Enquanto

21) Fim-Se

Figura 5.13 - Algoritmo 3 - Troca Simultânea

Procedimento V:	
16.1)	Se ($Rota_{Nova}$ é viável) Então
16.2)	Se ($Custo_{Novo} < Custo_{Melhor}$) Então
16.3)	Defina $Custo_{Melhor} = Custo_{Novo}$
16.4)	Defina $Rota_{Melhor} = Rota_{Nova}$
16.5)	Fim-Se
16.6)	Fim-Se

Figura 5.14 - Procedimento V

Através da combinação do algoritmo 3 (utilizado na fase de construção) e do algoritmo 4 (utilizado na fase de busca local), obtemos então o algoritmo GR2:

Algoritmo GR2: Vizinho mais Próximo + Troca Simultânea	
1)	Defina $Custo_{Final} = \infty$
2)	Para $k = 1$ até 1000 Faça
3)	Aplique o algoritmo de construção (Algoritmo 3) para obter ($Rota_{Inicial}$, $Custo_{Inicial}$)
4)	Aplique a busca local (Algoritmo 4), gerando uma nova solução ($Rota_{Melhor}$, $Custo_{Melhor}$)
5)	Se ($Custo_{Melhor} < Custo_{Final}$) Então
6)	Defina $Custo_{Final} = Custo_{Melhor}$
7)	Defina $Rota_{Final} = Rota_{Melhor}$
8)	Fim-Se
9)	Fim-Para
10)	Defina Solução-GRASP = $Rota_{Final}$

Figura 5.15 - Algoritmo GR2

5.5.5 - Algoritmo 5 - Filtro

Ao invés de gerarmos apenas uma $Rota_{Inicial}$ a partir do algoritmo 1 ou do algoritmo 3, executamos m vezes estes algoritmos (neste trabalho, definimos $m = 50$) selecionando a melhor $Rota_{Inicial}$. Em seguida, a partir desta rota, aplicamos um dos algoritmos utilizados na fase de busca local.

Algoritmo 5: Filtro
1) Defina $Custo_{Provisório} = \infty$
2) Para $j = 1$ até 50 Faça
3) Execute o algoritmo da fase de construção (Algoritmo 1 ou 3) para obter $(Rota_{Inicial}$, $Custo_{Inicial})$
4) Se $(Custo_{Inicial} < Custo_{Provisório})$ Então
5) Defina $Custo_{Provisório} = Custo_{Inicial}$
6) Defina $Rota_{Provisória} = Rota_{Inicial}$
7) Fim-Se
8) Fim-Para
9) Defina $Custo_{Inicial} = Custo_{Provisório}$
10) Defina $Rota_{Inicial} = Rota_{Provisória}$

Figura 5.16 - Algoritmo 5 - Filtro

Utilizando o algoritmo 5 combinado com os algoritmos $GR1$ e $GR2$, obtemos os algoritmos $GR1F$ e $GR2F$ apresentados a seguir:

Algoritmo GR1F: Vizinho mais Próximo + VNS

1) Defina $Custo_{Final} = \infty$

2) Para $k = 1$ até 1000 Faça

3) Aplique o Algoritmo 5 (considerando o Algoritmo 1) para obter $(Rota_{Inicial}, Custo_{Inicial})$

4) Aplique a busca local (Algoritmo 2) gerando uma nova solução $(Rota_{Melhor}, Custo_{Melhor})$

5) Se $(Custo_{Melhor} < Custo_{Final})$ Então

6) Defina $Custo_{Final} = Custo_{Melhor}$ e $Rota_{Final} = Rota_{Melhor}$

7) Fim-Se

8) Fim-Para

9) Defina Solução-GRASP $= Rota_{Final}$

Figura 5.17 - Algoritmo GR1F

Algoritmo GR2F: Vizinho mais Próximo + Troca Simultânea

1) Defina $Custo_{Final} = \infty$

2) Para $k = 1$ até 1000 Faça

3) Aplique o Algoritmo 5 (considerando o Algoritmo 3) para obter $(Rota_{Inicial}, Custo_{Inicial})$

4) Aplique a busca local (Algoritmo 4) gerando uma nova solução $(Rota_{Melhor}, Custo_{Melhor})$

5) Se $(Custo_{Melhor} < Custo_{Final})$ Então

6) Defina $Custo_{Final} = Custo_{Melhor}$ e $Rota_{Final} = Rota_{Melhor}$

8) Fim-Se

9) Fim-Para

10) Defina Solução-GRASP $= Rota_{Final}$

Figura 5.18 - Algoritmo GR2F

5.5.6 - Algoritmo 6 - Reconexão por Caminhos

O algoritmo 6 realiza o religamento (reconexão) entre as duas melhores soluções distintas que foram armazenadas no Conjunto Elite. A partir deste conjunto, definimos uma solução como a solução base (S_B) e a outra como a solução guia (S_G).

O objetivo deste algoritmo é o de transformar a solução base (S_B) na solução guia (S_G) através de trocas de vértices que estão em S_B por vértices que estão em S_G.

Inicialmente, definimos a solução intermediária inicial igual a solução base. Observando que em cada troca, construímos uma nova solução intermediária (S_I) (nova rota), considerando um dos seguintes casos:

1^o Caso: Se $t_i \in T$ está na k-ésima posição de S_I e $t_j \in T$ está na k-ésima posição de S_G - Trocamos t_i por t_j em S_I e em seguida, trocamos t_j que está na p-ésima posição de S_I por t_i ($p \neq k$).

2^o Caso: Se $t_i \in T$ está na k-ésima posição de S_I e $v_j \in V \backslash T$ está na k-ésima posição de S_G - Trocamos t_i por v_j em S_I e verificamos se v_j já pertencia a S_I na p-ésima posição ($p \neq k$). Caso isto ocorra, então substituímos este vértice v_j (p-ésima posição) por t_i. Em caso contrário, inserimos o vértice t_i no final de S_I.

3^o Caso: Se $v_j \in V \backslash T$ está na k-ésima posição de S_I e $t_i \in T$ está na k-ésima posição de S_G - Trocamos v_j por t_i em S_I e excluímos t_i da p-ésima posição de S_I ($p \neq k$). Se a rota associada a S_I se tornar inviável após estas duas operações, inserimos o vértice v_j na p-ésima posição de S_I.

4^o Caso: Se $v_j \in V \backslash T$ está na k-ésima posição de S_I e $v_i \in V \backslash T$ está na k-ésima posição de S_G - Trocamos v_j por v_i em S_I. Se o vértice v_i não pertencia a S_I e após esta troca a rota se tornar inviável, então inserimos o vértice v_j no final de S_I.

Se o vértice v_i estava na p-ésima posição de S_I, excluímos v_i desta posição. Se após esta troca, a rota se tornar inviável, então inserimos v_j na p-ésima posição de S_I.

A seguir, apresentamos um exemplo onde são efetuadas as trocas dos vértices considerando estes quatro casos:

Passo 0: Definimos a solução base, a solução guia e a solução intermediária inicial

$$S_B = t_1\ t_2\ v_2\ v_3\ t_3 \quad , \quad S_G = t_3\ v_1\ t_2\ v_2\ t_1 \quad , \quad S_I = S_B$$

Passo 1: Observando que t_1 está na 1^a posição de S_I e t_3 está na 1^a posição de S_G (caso 1), trocamos t_1 por t_3 na 1^a posição de S_I. Em seguida, substituímos t_3 na 5^a posição de S_I por t_1.

$S_I = \underbrace{t_3}\ t_2\ v_2\ v_3\ t_1$ (1^a solução intermediária)

$S_G = \underbrace{t_3}\ v_1\ t_2\ v_2\ t_1$

Passo 2: Observando que t_2 está na 2^a posição de S_I e v_1 está na 2^a posição de S_G (caso 2), trocamos t_2 por v_1 na 2^a posição de S_I e inserimos t_2 no final de S_I, pois v_1 não estava em S_I.

$S_I = t_3\ \underbrace{v_1}\ v_2\ v_3\ t_1\ t_2$ (2^a solução intermediária)

$S_G = t_3\ \underbrace{v_1}\ t_2\ v_2\ t_1$

Passo 3: Observando que v_2 está na 3^a posição de S_I e t_2 está na 3^a posição de S_G (caso 3), trocamos v_2 por t_2 na 3^a posição de S_I e em seguida, retiramos t_2 da 6^a posição de S_I. Supondo que a rota associada a S_I é viável, não há necessidade de inserir v_2 na posição de t_2 (6^a posição).

$S_I = t_3\ v_1\ \underbrace{t_2}\ v_3\ t_1$ (3^a solução intermediária)

$S_G = t_3\ v_1\ \underbrace{t_2}\ v_2\ t_1$

Passo 4: Observando que v_3 está na 4^a posição de S_I e v_2 está na 4^a posição de S_G (caso 4), trocamos v_3 por v_2 na 4^a posição de S_I. Como v_2 não estava em S_I e a rota associada a S_I é supostamente viável, não há necessidade inserir o vértice v_3 em S_I.

$S_I = t_3\ v_1\ t_2\ \underbrace{v_2}\ t_1$ (4^a solução intermediária) , $S_G = t_3\ v_1\ t_2\ \underbrace{v_2}\ t_1$

Neste último passo, obtemos a solução intermediária S_I igual a solução guia, concluindo então o processo de reconexão.

Se tivermos a solução base maior do que a solução guia, $|S_B| > |S_G|$, efetuamos no máximo k trocas de vértices, sendo $k = |S_G|$. Além disso, para a r-ésima solução intermediária S_I $(r = k)$, desconsideramos os vértices que estão entre a posição $k+1$ e a última posição de S_B, obtendo $S_I = S_G$.

Se $|S_B| < |S_G|$, efetuamos no máximo k trocas de vértices, sendo $k = |S_B|$. Além disso, inserimos na r-ésima solução intermediária S_I $(r = k)$, os vértices que estão entre a posição $k+1$ e a última posição de S_G e novamente temos $S_I = S_G$.

Algoritmo 6: Reconexão por Caminhos

1) Defina S_B e S_G a partir do $Conjunto_{Elite}$ e Defina $k = 0$

2) Defina $S_I = S_B$ e Defina $Tot_{Elementos} = Min(|S_I|, |S_G|)$

3) Enquanto ($Tot_{elementos} > k$) Faça

4) Defina $k = k + 1$

5) Defina $e_G = S_G[k]$ e Defina $e_I = S_I[k]$

6) Se ($e_G \neq e_I$) Então

7) Se ($e_I \in T$ e $e_G \in T$) Então Execute Procedimento Caso1

8) Se ($e_I \in T$ e $e_G \in V \backslash T$) Então Execute Procedimento Caso2

9) Se ($e_I \in V \backslash T$ e $e_G \in T$) Então Execute Procedimento Caso3

10) Se ($e_I \in V \backslash T$ e $e_G \in V \backslash T$) Então Execute Procedimento Caso4

11) Atualize o $Conjunto_{Elite}$ considerando o custo da rota associada a S_I

12) Fim-Se

13) Fim-Enquanto

Figura 5.19 - Algoritmo 6 - Reconexão por Caminhos

No passo (1), definimos a solução base e a solução guia a partir das rotas do $Conjunto_{Elite}$. No passo (2) definimos a primeira solução intermediária (S_I) igual a solução base (S_B) e definimos em $Tot_{Elementos}$ o total de posições que serão verificadas em S_I e S_G . O algoritmo termina após a troca de k vértices de S_I por S_G (passo (3)).

No passo (4) incrementamos a variável k para acessar uma nova posição de S_I e S_G . No passo (5), associamos a e_G o vértice que está na k-ésima posição de S_G e associamos a e_I o vértice que está na k-ésima posição de S_I .

Se o vértice e_I for diferente do vértice e_G (passo (6)), verificamos a que conjuntos e_I e e_G pertencem (conjuntos associados ao Problema de Recobrimento

de Rotas). Em seguida, executamos um dos procedimentos associados aos casos descritos na página 65 (passos 7, 8, 9 ou 10).

No passo (11), substituímos uma das rotas que estão no $Conjunto_{Elite}$ pela rota (solução intermediária S_I) obtida na k-ésima iteração, se o custo desta rota for menor do que o da rota de maior custo que está no $Conjunto_{Elite}$.

Com a utilização do algoritmo 6, obtemos os algoritmos GR1FPR-V0 e GR1FPR-V1, observando que estes algoritmos diferem basicamente pela forma de construção do $Conjunto_{Elite}$ e pelo "momento" em que executamos o Algoritmo 6. Nos dois algoritmos, consideramos $|Conjunto_{Elite}| = 2$.

Procedimento Caso1
7.1) Defina $S_I[p] = e_I$ ($p =$ posição que e_G ocupa em S_I)
7.2) Defina $S_I[k] = e_G$

Figura 5.20 - Procedimento Caso1

Procedimento Caso2
8.1) Se (e_G está na p-ésima posição de S_I) ($p \neq k$)
8.2) Então Defina $S_I[p] = e_I$
8.3) Senão Insira e_I no final de S_I e Defina $Tot_{Elementos} = Min(\|S_I\|, \|S_G\|)$
8.4) Fim-Se
8.5) Defina $S_I[k] = e_G$

Figura 5.21 - Procedimento Caso2

Procedimento Caso3
9.1) Exclua e_G da p-ésima posição de S_I ($p \neq k$)
9.2) Defina $S_I[k] = e_G$
9.3) Se (S_I está associada a uma rota inviável)
9.4) Então Defina $S_I[p] = e_I$
9.5) Senão Defina $Tot_{Elementos} = Min(\|S_I\|, \|S_G\|)$
9.6) Fim-Se

Figura 5.22 - Procedimento Caso3

Procedimento Caso4
10.1) Se (e_G está na p-ésima posição de S_I) Então Exclua e_G desta posição
10.2) Defina $S_I[k] = e_G$
10.3) Se (S_I está associada a uma rota inviável) Então
10.4) Se (passo (10.1) foi executado)
10.5) Então Defina $S_I[p] = e_I$
10.6) Senão Insira e_I no final de S_I
10.7) Defina $Tot_{Elementos} = \text{Min } (\|S_I\|, \|S_G\|)$
10.8) Fim-Se

Figura 5.23 - Procedimento Caso4

Utilizando o algoritmo 6 combinado com os algoritmos do vizinho mais próximo, de Filtro e o VNS, obtivemos os algoritmos GR1FPR-V0 e GR1FPR-V1 apresentados a seguir.

Algoritmo GR1FPR-V0: Viz.mais Próximo, Filtro, Reconexão por Caminhos e VNS

1) Defina $Custo_{Final} = \infty$

2) Defina $Conjunto_{Elite} = \emptyset$

3) Para $k = 1$ até 1000 Faça

4) Se (k *mod* $50 = 0$)

5) Então Aplique o Algoritmo 5 (considerando o Algoritmo 1) para obter ($Rota_{Inicial1}$, $Custo_{Inicial1}$) e ($Rota_{Inicial2}$, $Custo_{Inicial2}$)

6) Defina $Conjunto_{Elite} = \{Rota_{Inicial1}$, $Rota_{Inicial2}\}$

7) Execute o Algoritmo 6 considerando a melhor solução do $Conjunto_{Elite}$ ($Rota_{Inicial}$, $Custo_{Inicial}$)

8) Senão

9) Aplique o Algoritmo 5 (considerando o Algoritmo 1) para obter ($Rota_{Inicial}$, $Custo_{Inicial}$)

10) Fim-Se

11) Aplique a busca local (Algoritmo 2), gerando uma nova solução ($Rota_{Melhor}$, $Custo_{Melhor}$)

12) Se ($Custo_{Melhor} < Custo_{Final}$) Então

13) Defina $Custo_{Final} = Custo_{Melhor}$

14) Defina $Rota_{Final} = Rota_{Melhor}$

15) Fim-Se

16) Fim-Para

17) Defina Solução-GRASP = $Rota_{Final}$

Figura 5.24 - Algoritmo GR1FPR-V0

No passo (1) definimos o $Custo_{Final}$ e no passo (2) inicializamos o $Conjunto_{Elite}$.

A cada 50 iterações deste algoritmo (passo (4)) aplicamos o Algoritmo 5 (Filtro) considerando o Algoritmo 1. Porém, ao invés de guardarmos apenas uma $Rota_{Inicial}$ como nos quatro algoritmos GRASP (GR1,GR2, GR1F e GR2F), guardamos as duas melhores rotas ($Rota_{Inicial1}$, $Rota_{Inicial2}$) (passo (5)). No passo (6) definimos o $Conjunto_{Elite}$ com estas duas rotas obtidas na fase de construção.

No passo (7) executamos o Algoritmo 6 utilizando as rotas que estão no $Conjunto_{Elite}$, escolhendo de forma aleatória a solução base e a solução guia. E após a execução do algoritmo 6, consideramos como $Rota_{Inicial}$ a solução de menor custo, presente no conjunto $Conjunto_{Elite}$.

Se estivermos em uma iteração k tal que ($k \ mod \ 50 \ \neq 0$) , aplicamos o Algoritmo 5 (Filtro) e obtemos a $Rota_{Inicial}$ (passo (9)).

Após a obtenção da $Rota_{Inicial}$, aplicamos no passo (11) o algoritmo de busca local (Algoritmo 2).

Se $Custo_{Melhor}$ for menor que $Custo_{Final}$ (passo (12)), então, no passo (13) e (14) atualizamos $Custo_{Final}$ e a $Rota_{Final}$.

Algoritmo GR1FPR-V1: Viz.mais Próximo, Filtro, Reconexão por Caminhos + VNS
1) Defina $Custo_{Final} = \infty$
2) Defina $Conjunto_{Elite} = \emptyset$
3) Para $k = 1$ até 1000 Faça
4) Aplique o Algoritmo 5 (considerando o Algoritmo 1) para obter $(Rota_{Inicial}, Custo_{Inicial})$
5) Aplique a busca local (Algoritmo 2), gerando uma nova solução $(Rota_{Melhor}, Custo_{Melhor})$
6) Atualize a $Conjunto_{Elite}$ considerando a $(Rota_{Melhor})$
7) Se (k *mod* $50 = 0$) Então
8) Execute o Algoritmo 6 definindo a 1^a *Rota* do $Conjunto_{Elite}$ como S_B e a 2^a rota como S_G
9) Execute o Algoritmo 6 considerando a 1^a *Rota* do $Conjunto_{Elite}$ como S_G e a 2^a rota como S_B
10) Fim-Se
11) Fim-Para
12) Defina $Rota_{Final}$ = Rota de menor custo do $Conjunto_{Elite}$
13) Defina Solução-GRASP = $Rota_{Final}$

Figura 5.25 - Algoritmo GR1FPR-V1

No passo (1) definimos o $Custo_{Final}$ e no passo (2) inicializamos o $Conjunto_{Elite}$. Diferentemente do algoritmo GR1FPR-V0, o $Conjunto_{Elite}$ será sempre atualizado após a aplicação do Algoritmo 2 e do Algoritmo 6.

No passo (4) aplicamos o Algoritmo 5 (considerando o Algoritmo 1) para obter a $Rota_{Inicial}$. No passo (5) aplicamos o Algoritmo 2 (busca local), gerando uma nova rota $(Rota_{Melhor})$.

No passo (6) atualizamos o $Conjunto_{Elite}$ com a $Rota_{Melhor}$, se esta rota tem custo menor do que a rota de maior custo do $Conjunto_{Elite}$.

Se estivermos em uma iteração k tal que $k \; mod \; 50 \; = \; 0$ (passo (7)), então, executamos o Algoritmo 6 (passo (8)) definindo a primeira rota do $Conjunto_{Elite}$ como solução base e a segunda rota como solução guia.

No passo (9), executamos novamente o Algoritmo 6, considerando a primeira rota do $Conjunto_{Elite}$ como solução guia e a segunda rota como solução base e no passo (12) definimos a $Rota_{Final}$ como a rota de menor custo do $Conjunto_{Elite}$.

6 - Resultados Computacionais

6.1 - Introdução:

Neste capítulo, faremos uma exposição dos resultados numéricos / computacionais obtidos para o Problema de Recobrimento de Rotas considerando a utilização: da nova formulação matemática (seção 3.2), da relaxação e da heurística lagrangeana (seção 3.3), da regra nova de redução (capítulo 4) e dos algoritmos baseados nas metaheurísticas GRASP e VNS (capítulo 5).

O desenvolvimento deste capítulo está dividido em quatro seções: Na seção 6.2 temos uma descrição do programa de geração dos dados. Na seção 6.3 temos apresentação dos resultados computacionais obtidos a partir da utilização das regras de redução: nova, 3 e 4 (regras 3 e 4 foram apresentadas na literatura). Na seção 6.4 temos os resultados da nova formulação matemática considerando o grafo original e os grafos reduzidos. Concluindo nossa exposição, na seção 6.5, temos os resultados dos algoritmos GRASP e uma comparação da performance destes algoritmos.

Todos os programas implementados neste trabalho de tese, foram processados em um microcomputador modelo IBM-PC dotado de um processador Pentium IV (1.6 Mhz) com 256 MB de memória.

6.2 - Programa de Geração dos Dados:

De acordo com nossa pesquisa bibliográfica, não existem problemas teste para o PRR, disponíveis em bibliotecas públicas. Por esse motivo, desenvolvemos um programa em linguagem C para gerar um conjunto de problemas que serviram como base para testar a nova formulação, as regras de redução e os algoritmos GRASP .

Inicialmente, o programa solicita ao usuário que defina o número de vértices de T (subconjunto de vértices de V que obrigatoriamente devem fazer parte da rota), o número de vértices de $V \backslash T$ (conjunto de vértices que podem ou não fazer parte da rota) e o número de vértices de W (conjunto de vértices que devem ser cobertos pelos vértices que fazem parte da rota, ou seja, vértices de V).

No passo seguinte, o programa gera de forma aleatória (na tela), coordenadas (x, y) (considerando que $(1 \leq x \leq 1000)$ e $(1 \leq y \leq 1000))$. Estas coordenadas representam a localização dos vértices T , $V \backslash T$ e W .

Com base nestas coordenadas, são calculadas as distâncias euclidianas d_{ij} entre todos os vértices do conjunto N (d_{ij} representa a distância percorrida do vértice v_i para o vértice v_j na aresta $(v_i, v_j) \in E$) e a partir destas distâncias, o programa determina a distância de cobertura d que será utilizada na rota:

$$d = max \{ min_{w_l \in W} \{ d_{lk} \} \} , \ \forall \ w_l \in W \ , \forall \ v_k \in V$$

Com o cálculo desta distância d, garantimos que todos os vértices do conjunto W serão cobertos por pelo menos um vértice do conjunto V.

Considerando estes dados, o programa gera dois arquivos de saída: No primeiro temos uma matriz de distâncias $D = [d_{ij}]$ que está definida sobre E e no segundo arquivo, temos o total de vértices de T, $V\backslash T$ e W e o valor de d.

6.3 - Resultados Computacionais das Regras de Redução

Conforme observado no capítulo 4, as regras de redução têm como objetivo principal reduzir o número de vértices do grafo original associado ao PRR produzindo um grafo reduzido, de forma que o conjunto de soluções viáveis associadas a esse grafo, seja um subconjunto de soluções viáveis para o grafo original.

Com a utilização do grafo reduzido (dado de entrada) para a formulação matemática (nova formulação e relaxação lagrangeana) e para os algoritmos GRASP, há a possibilidade de resolver-se instâncias maiores do PRR e de obter soluções viáveis (limites superiores) de melhor qualidade.

De acordo com o capítulo 3, verificamos que as regras 1 e 2 da literatura podem produzir grafos reduzidos com soluções inviáveis para o grafo original. Desta forma, utilizamos somente as regras 3 e 4 da literatura e a regra nova (proposta neste trabalho) para produzir os grafos reduzidos. A seguir, descrevemos novamente estas tres regras:

Regra 3: Remover o vértice $w_l \in W$ do grafo original, se existe $t_i \in T$, tal que $\delta_{li} = 1$.

Regra 4: Remover o vértice $v_i \in V\backslash T$ do grafo original, se $\delta_{li} = 0, \quad \forall w_l \in W$.

Regra Nova: Se existe um único vértice $v_i \in V\backslash T$ tal que $\delta_{li} = 1$, $w_l \in W$, então o vértice v_i será transformado em um vértice do conjunto T, $v_i \rightarrow t_i \in T$.

Para obtenção dos grafos reduzidos a partir do grafo original, considerando as regras 3 e 4 da literatura e a regra nova (acima descritas), implementamos um programa em linguagem C que utiliza como dado de entrada os dois arquivos obtidos a partir do programa de geração de dados.

Neste programa, temos a visualização dos vértices dos conjuntos W e $V\backslash T$ que foram eliminados a partir da aplicação da regras 3 e 4 da literatura (a utilização destas duas regras define o conjunto 1) e dos vértices do conjunto $V\backslash T$ que foram transformados em vértices de T utilizando a regra nova e quantos vértices de W e $V\backslash T$ foram eliminados a partir da aplicação das regras 3 e 4, após a aplicação da regra nova (a utilização destas três regras define o conjunto 2).

Após a aplicação das regras destes dois conjuntos, geramos como dados de saída quatro arquivos associados aos grafos reduzidos (matriz de distâncias e total de vértices T, $V \backslash T$ e W e o valor de d).

Com a finalidade de analisar e comparar a eficiência dos dois conjuntos de regras de redução, geramos problemas teste associados a grafos de 10 até 500 vértices. E para cada número de vértices formamos grupos com 15 problemas teste, ou seja, 15 grafos com 10 vértices ($|V \cup W| = 10$), 15 grafos com 20 vértices ($|V \cup W| = 20$), e assim por diante.

Na tabela 6.1, apresentamos o número de vértices eliminados em média pelo conjunto 1 (regras 3 e 4 da literatura) $\bar{R}_1$ e pelo conjunto 2 (regra proposta neste trabalho mais as regras 3 e 4 da literatura) $\bar{R}_2$. Esta média é formada pelo número de vértices eliminados nos exemplos do mesmo grupo. Além da apresentação destas médias, temos o cálculo da diferença percentual (γ = desempenho) entre $\bar{R}_2$ e $\bar{R}_1$, considerando a seguinte expressão:

$$\gamma = 100.\frac{(\bar{R}_2 - \bar{R}_1)}{\bar{R}_2}$$

Tabela 6.1 - Comparação entre as regras de redução do conjunto 2 e do conjunto 1

Número de vértices do grafo original	Número de vértices eliminados em média pelo conj. 1 ($\bar{R}_1$)	Número de vértices eliminados em média pelo conj. 2 ($\bar{R}_2$)	Desempenho do conj.2 em relação ao conj. 1 (γ)
10	3,7	4,6	24,3%
20	8,0	10,1	26,3%
30	14,1	15,7	11,4%
40	18,6	21,1	13,4%
50	24,7	26,5	7,3%
60	30,2	34,0	12,6%
70	35,1	36,4	3,7%
80	43,3	47,4	9,5%
90	45,6	51,5	12,9%
100	53,3	57,0	6,9%
150	81,1	86,3	6,4%
200	99,1	105,3	6,3%
250	138,9	145,4	4,7%
300	168,2	172,8	2,7%
350	202,4	208,6	3,1%
400	230,1	239,9	4,3%
450	213,3	226,4	6,1%
500	298,6	310,7	4,1%
Média Geral	94,9	99,9	9,2

Analisando a tabela 6.1, observamos que em todos os casos, com a aplicação das regras do conjunto 2 obtivemos melhores médias, isto é, houve uma maior redução no número de vértices do grafo original, considerando as regras nova, 3 e 4.

Nos gráficos 6.1 e 6.2 apresentados abaixo, temos respectivamente 30 exemplos de problemas de 50 vértices e de problemas de 100 vértices. Considerando o número de vértices retirados do grafo original após a aplicação das regras dos conjuntos 1 e 2.

Verificamos através destes gráficos, que na maioria dos problemas, com aplicação das regras do conjunto 2, geramos grafos com o número de vértices menor do que os grafos reduzidos obtidos a partir da utilização das regras do conjunto 1.

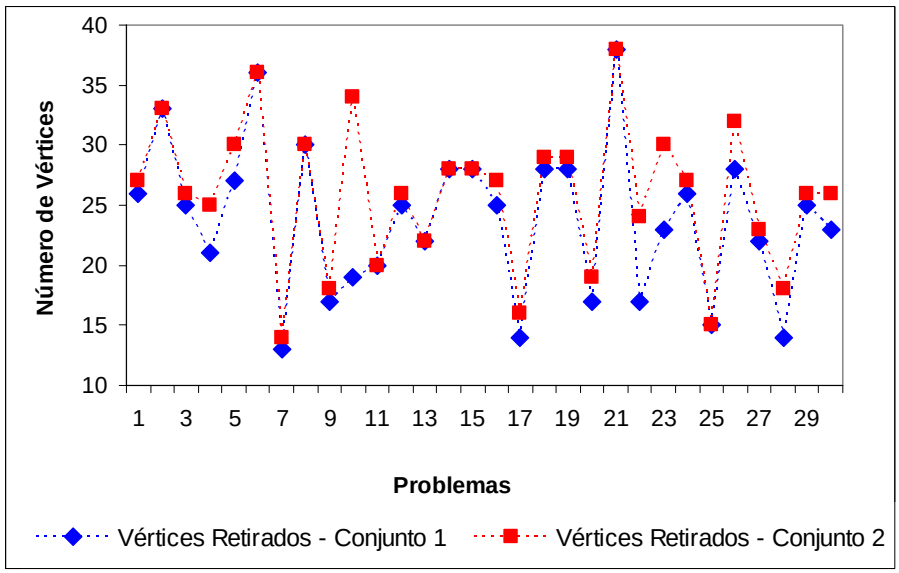

Gráfico 6.1 - Comparação do número de vértices retirados pelos conjuntos 1 e 2, considerando 30 problemas de 50 vértices

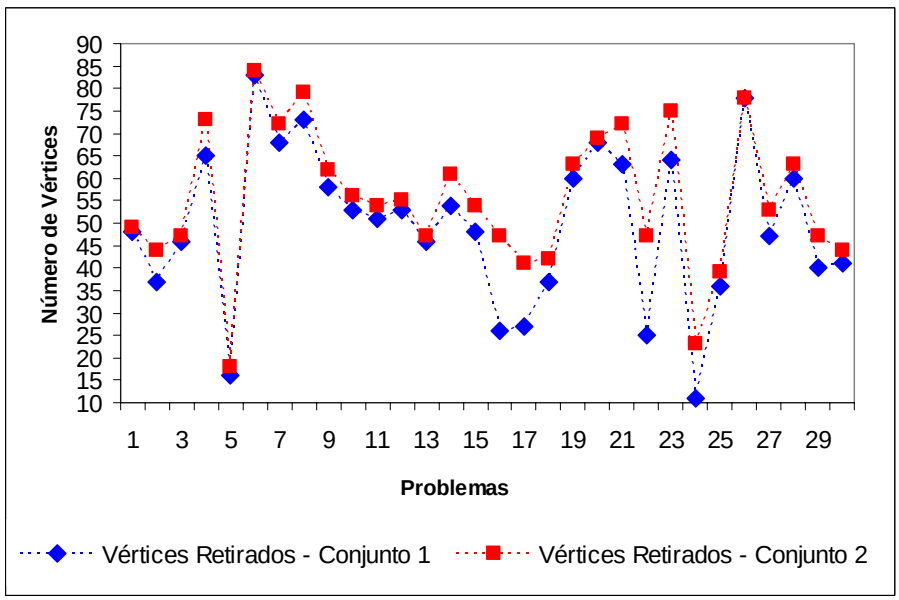

Gráfico 6.2 - Comparação do número de vértices retirados pelos conjuntos 1 e 2, considerando 30 problemas de 100 vértices

6.4 - Resultados da Formulação Matemática e da Relaxação Lagrangeana

6.4.1 - Introdução

Nesta seção, temos a apresentação dos resultados computacionais obtidos a partir da utilização da nova formulação matemática que foi implementada através do pacote de otimização XPRESS[39].

Conforme apresentado no capítulo 3, esta formulação se diferencia das demais formulações existentes na literatura quanto às restrições que asseguram a não formação de ciclos desconexos da origem.

Além dos resultados da nova formulação, temos:

- A apresentação dos resultados da relaxação e da heurística lagrangeana, que foram implementadas utilizando o pacote de otimização XPRESS[39] e a linguagem de programação Delphi 6.0. Nesta relaxação, agregamos à função objetivo da nova formulação as restrições que garantem a cobertura dos vértices do conjunto W (vide seção 3.3).

- Um quadro comparativo entre a nova formulação proposta e a formulação de Maniezzo [8].

6.4.2 - Resultados da Nova Formulação

Nas tabelas 6.2 e 6.3 apresentadas a seguir, temos os resultados computacionais para um total de 70 problemas teste que foram obtidos a partir do programa de geração de dados.

Nas colunas de 1 a 4 destas duas tabelas temos o nome dos problemas teste (são nomeados de forma que o último algarismo está associado ao número do problema e os demais algarismos estão associados ao número de vértices de $V \cup W$) e os valores (V_O) , (V_L) e (Gap_{OL}) associados respectivamente à solução ótima, à solução da relaxação linear e ao gap entre a solução ótima e a solução da relaxação linear (considerando o grafo original).

Nas colunas de 5 a 7 temos os tempos de processamento T_O , T_{R1} e T_{R2} (em segundos) da nova formulação, considerando respectivamente o grafo original e os grafos reduzidos pelos conjuntos 1 e 2 (estes dois grafos reduzidos foram obtidos a partir do programa que aplica as regras do conjunto 1 e do conjunto 2).

Nas colunas 8, 9 e 10 temos respectivamente o valor da melhor solução viável (V_G) , obtida a partir de um dos algoritmos GRASP, o gap entre a solução viável

e a solução da relaxação linear (Gap_{GL}) e o gap entre a solução viável e a solução ótima (Gap_{GO}) .

Com base nestas tabelas, podemos fazer as seguintes considerações:

- Em 37% dos casos, considerando o grafo original, foi possível obter a solução ótima no tempo limite de 8 horas. E quando consideramos os grafos reduzidos pelos conjuntos 1 e 2, tivemos percentuais respectivamente de 52.8% e de 58.6% .

- Para maioria dos problemas teste com 60 ou mais vértices, não foi possível obter a solução ótima no tempo limite de 8 horas, quando considerado o grafo original.

- A obtenção da solução ótima para a maioria dos problemas teste com número de vértices maior ou igual a 60, só foi possível a partir da utilização dos grafos reduzidos pelos conjuntos 1 e 2.

- Considerando os problemas para os quais se obteve o valor ótimo, o gap (Gap_{OL}) entre o valor da solução ótima e o valor da relaxação linear foi em média de $43,21\%$.

- Ainda considerando o mesmo conjunto de problemas, observamos que o gap entre o valor da melhor solução dos algoritmos GRASP e a solução ótima (Gap_{GO}) foi em média de $0,70\%$, o que indica que os algoritmos GRASP produzem soluções razoáveis.

- O gap entre o valor da melhor solução dos algoritmos GRASP e o valor da relaxação linear, foi em média de $43,79\%$.

Embora Gendreau e Maniezzo tenham proposto formulações matemáticas para o PRR, todos os resultados computacionais apresentados em seus trabalhos, são baseados na utilização de heurísticas.

E no caso deste trabalho, utilizando uma nova formulação, conseguimos obter a solução ótima para todos os problemas teste com até 50 vértices.

Tabela 6.2 - Resultados da Nova Formulação (Problemas de 10 a 70 vértices)

Prob.	Formulação						Algoritmo		
Teste	V_O	V_L	Gap_{OL}	T_O	T_{R1}	T_{R2}	V_G	Gap_{GL}	Gap_{GO}
101	1129,29	674,29	67,48%	1	0	0	1129,29	67,48%	0,00%
102	730,33	654,97	11,51%	0	0	1	730,33	11,51%	0,00%
103	887,55	773,71	14,71%	1	0	0	887,56	14,71%	0,00%
104	1257,69	784,90	60,24%	0	0	1	1257,69	60,24%	0,00%
105	1360,32	608,91	123,40%	1	1	0	1360,32	123,40%	0,00%
201	1920,24	1382,67	38,88%	1	1	1	1920,25	38,88%	0,00%
202	1490,84	1061,95	40,39%	2	1	1	1490,85	40,39%	0,00%
203	1493,43	964,45	54,85%	2	1	1	1493,43	54,85%	0,00%
204	995,34	883,32	12,68%	0	0	0	995,34	12,68%	0,00%
205	1721,00	992,24	73,45%	3	1	0	1721,00	73,45%	0,00%
301	1983,07	1480,20	33,97%	3	1	1	1983,07	33,97%	0,00%
302	1883,95	1493,72	26,12%	1	1	0	1883,95	26,12%	0,00%
303	1966,87	1254,76	56,75%	5	1	1	1966,87	56,75%	0,00%
304	1865,63	1565,68	19,16%	2	1	0	1865,63	19,16%	0,00%
305	1516,74	1143,53	32,64%	7	1	1	1516,74	32,64%	0,00%
401	1691,38	1173,57	44,12%	84	0	0	1691,39	44,12%	0,00%
402	2113,80	1737,67	21,65%	140	5	6	2113,80	21,65%	0,00%
403	2364,55	1883,38	25,55%	14	3	4	2364,55	25,55%	0,00%
404	1986,96	1780,33	11,61%	1	1	1	1986,96	11,61%	0,00%
405	1835,31	997,08	84,07%	5	1	1	1835,31	84,07%	0,00%
501	2101,26	1377,58	52,53%	60	2	3	2101,26	52,53%	0,00%
502	1915,40	1065,21	79,81%	2880	253	151	1970,53	84,99%	2,80%
503	2371,39	1798,52	31,85%	201	9	7	2371,39	31,85%	0,00%
504	1853,10	1484,00	24,87%	10	3	4	1853,10	24,87%	0,00%
505	2171,61	1697,98	27,89%	12	4	2	2171,61	27,89%	0,00%
601	2229,30	1655,41	34,67%	****	44	43	2245,99	35,68%	0,74%
602	1992,16	1101,53	80,85%	****	2	3	1992,16	80,85%	0,00%
603	1678,43	1098,60	52,78%	****	215	198	1678,43	52,78%	0,00%
604	1804,34	1143,70	57,76%	97	2	1	1851,60	63,35%	2,55%
605	2243,88	1468,75	52,77%	****	18501	15291	2243,88	52,77%	0,00%
701	2335,17	1531,08	52,52%	****	****	4256	2479,57	61,95%	5,82%
702	2537,24	1925,84	31,75%	****	17	24	2537,24	31,75%	0,00%
703	2629,01	2196,06	19,71%	****	8	6	2632,93	19,89%	0,15%
704	**	2112,10		****	****	****	2883,66	36,53%	
705	1702,08	1121,03	51,83%	****	****	12	1702,08	51,83%	0,00%

** Solução ótima não foi encontrada no tempo de 8 horas

**** Tempo de 8 horas ultrapassado (sem obtenção de solução ótima)

Tabela 6.3 - Resultados da Nova Formulação (Problemas de 80 a 500 vértices)

Prob.	Formulação						Algoritmo		
Teste	V_O	V_L	Gap_{OL}	T_O	T_{R1}	T_{R2}	V_G	Gap_{GL}	Gap_{GO}
801	2303,89	1688,71	36,43%	****	45	37	2325,98	37,74%	0,95%
802	**	2041,31		****	****	****	2771,51	35,77%	
803	**	2182,67		****	****	****	2947,77	35,05%	
804	2312,83	1569,18	47,39%	****	371	57	2316,60	47,63%	0,16%
805	2343,53	1911,16	22,62%	****	6	5	2343,53	22,62%	0,00%
901	**	1415,30		****	****	****	2666,92	88,43%	
902	**	2504,19		****	****	****	3162,35	26,28%	
903	**	2131,33		****	****	****	2904,93	36,30%	
904	**	1654,06		****	****	****	2256,41	36,42%	
905	**	2564,86		****	****	****	3180,89	24,02%	
1001	**	1704,74		****	****	****	2610,22	53,12%	
1002	2308,79	1658,01	39,25%	****	5	3	2308,79	39,25%	0,00%
1003	**	2126,18		****	****	****	2832,74	33,23%	
1004	2604,09	1629,32	59,83%	****	2270	1987	2632,59	61,58%	1,08%
1005	2496,25	1813,86	37,62%	****	****	1420	2500,41	37,85%	0,17%
2001	**	2417,48		****	****	****	3556,67	47,12%	
2002	2803,83	2271,07	23,46%	****	****	1260	2814,37	23,92%	0,37%
2003	**	2558,25		****	****	****	3752,91	46,70%	
2004	**	2298,99		****	****	****	3512,77	52,80%	
2005	**	2924,54		****	****	****	4393,59	50,23%	
3001	**	3103,12		****	****	****	4945,34	59,37%	
3002	**	2125,02		****	****	****	4179,68	96,69%	
3003	**	3987,74		****	****	****	5503,75	38,02%	
3004	**	4129,07		****	****	****	5671,74	37,36%	
3005	**	3133,69		****	****	****	4638,68	48,03%	
4001	**	3895,20		****	****	****	6342,74	62,83%	
4002	**	3356,29		****	****	****	5332,00	58,87%	
4003	**	4399,42		****	****	****	6484,45	47,39%	
4004	**	3154,19		****	****	****	5078,11	61,00%	
4005	**	2791,87		****	****	****	4963,06	77,77%	
5001	**	4158,44		****	****	****	6667,05	60,33%	
5002	**	2685,94		****	****	****	5496,85	104,65%	
5003	**	5522,13		****	****	****	8126,09	47,16%	
5004	**	4326,71		****	****	****	6637,34	53,40%	
5005	**	3077,88		****	****	****	5593,52	81,73%	

** Solução ótima não foi encontrada no tempo de 8 horas

**** Tempo de 8 horas ultrapassado (sem obtenção de solução ótima)

Com finalidade de estabelecer uma comparação entre a nova formulação proposta neste trabalho e a formulação proposta por Maniezzo *et al.* (vide seção 2.4.2), apresentamos na tabela 6.4, um conjunto de 20 problemas teste, considerando o número de vértices dos conjuntos V e W variado.

Nas colunas 2, 3, e 4 desta tabela, temos respectivamente: o número de restrições, número de variáveis binárias e o tempo de processamento da nova formulação. De forma análoga, temos nas colunas 5, 6, 7: o número de restrições, o número de variáveis binárias e o tempo de processamento da formulação proposta por Maniezzo (Maniezzo *et al.* [8]).

Podemos observar, que em 16 dentre os 20 problemas teste, o tempo de processamento da nova formulação foi razoavelmente inferior ao tempo de processamento da formulação de Maniezzo. Desta forma, há uma boa indicação de que a nova formulação proposta pode ser mais eficiente do que a formulação de Maniezzo.

Tabela 6.4 - Comparação entre a Nova Formulação e a Formulação de Maniezzo

Prob. Teste	(NF)			(FM)		
	Restrições	Var. Binárias	Tempo*	Restrições	Var. Binárias	Tempo*
108	108	49	0	109	52	0
208	407	196	2	407	106	2
306	1270	625	4	1103	460	1
309	227	100	1	570	445	138
407	837	400	5	1235	800	3
506	1287	625	255	1910	1250	2138
571	1657	784	76	5652	4781	247
607	1483	729	43	1185	433	405
608	1842	900	26	2740	1800	2534
609	2088	1024	230	2864	1802	20640
706	1168	576	43	992	402	115
709	1852	900	89	3395	2445	442
761	2372	1156	1750	4098	2884	27328
808	2120	1024	88	4286	3192	19624
809	1416	676	58	3924	3186	7176
881	1639	784	5	4709	3856	78
907	1432	676	50	4785	4031	228
981	1657	784	76	5652	4781	247
1007	1767	841	230	5939	4979	2060
1008	1059	484	3	5545	4972	10

(NF) Nova Formulação

(FM) Formulação de Maniezzo

$*$ Tempo em segundos

6.4.3 Resultados da Relaxação e da Heurística Lagrangeana

Na tabela 6.5 abaixo, apresentamos os resultados da relaxação lagrangeana e da heurística lagrangeana, considerando um conjunto de 12 problemas teste.

Nesta tabela, temos segundo a ordem das colunas: O nome do problema teste, o valor da melhor solução viável obtida a partir da utilização da heurística lagrangeana, o melhor limite inferior obtido na relaxação lagrangena, o melhor valor da relaxação linear do problema, o valor da solução ótima e o tempo de processamento da relaxação combinada com a heurística lagrangeana.

E nas colunas seguintes, temos a diferença percentual (Gap's) entre os valores: (1) do ótimo e da relaxação lagrangeana $(Gap_{(VO)/(RLA)})$, (2) do ótimo e da relaxação linear $(Gap_{(VO)/(RLI)})$, (3) da heurística lagrangeana e da relaxação lagrangeana $(Gap_{(HLA)/(RLA)})$, (4) da heurística lagrangeana e o ótimo $(Gap_{(HLA)/(VO)})$ e (5) da relaxação lagrangeana e da relaxação linear $(Gap_{(RLA)/(RLI)})$.

Tabela 6.5 - Valores da Heurística,das Relaxações Lagrangeana e Linear e Ótimo

Prob. Teste	Heurística Lagrangeana (HLA)	Relaxação. Lagrangeana (RLA)	Relaxação Linear (RLI)	Valor ótimo (VO)	Tempo (seg)
201	1947,21	1910,75	1382,67	1920,24	20
202	1553,87	1490,84	1061,95	1490,84	7
303	1967,49	1952,99	1254,76	1966,87	13
304	1920,13	1865,63	1565,68	1865,63	2
305	1757,90	1516,74	1143,53	1516,74	20
403	2419,98	2329,13	1883,38	2364,55	15
404	2689,38	1986,96	1780,33	1986,96	10
405	1986,63	1631,04	997,08	1835,31	12
501	2365,58	2079,15	1377,58	2101,26	4
504	1874,94	1827,16	1484,00	1853,10	12
505	2172,17	2170,06	1697,98	2171,61	8
604	3285,66	1776,52	1143,70	1804,34	143

$Gap_{(VO)/(RLA)}$	$Gap_{(VO)/(RLI)}$	$Gap_{(HLA)/(RLA)}$	$Gap_{(HLA)/(VO)}$	$Gap_{(RLA)/(RLI)}$
0,50%	38,88%	1,91%	1,40%	38,19%
0,00%	40,39%	4,23%	4,23%	40,39%
0,71%	56,75%	0,74%	0,03%	55,65%
0,00%	19,16%	2,92%	2,92%	19,16%
0,00%	32,64%	15,90%	15,90%	32,64%
1,52%	25,55%	3,90%	2,34%	23,67%
0,00%	11,61%	35,35%	35,35%	11,61%
12,52%	84,07%	21,80%	8,24%	63,58%
1,06%	52,53%	13,78%	12,58%	50,93%
1,42%	24,87%	2,61%	1,18%	23,12%
0,07%	27,89%	0,10%	0,03%	27,80%
1,57%	57,76%	84,95%	82,10%	55,33%

Com base nesta tabela, podemos fazer as seguintes considerações:

• Analisando os valores de $Gap_{(VO)/(RLA)}$ e $Gap_{(VO)/(RLI)}$, verificamos que utilizando a relaxação lagrangeana obtivemos limites inferiores bem melhores do que os limites obtidos a partir da relaxação linear. Ou seja, os valores da relaxação lagrangeana estão mais próximos do valores da solução ótima.

• A diferença percentual entre a solução ótima e a solução da relaxação lagrangeana $(Gap_{(VO)/(RLA)})$ foi em média de $1,61\%$. No caso da solução ótima e da solução da relaxação linear, esta diferença $(Gap_{(VO)/(RLI)})$ foi em média de $39,34\%$.

• O gap entre os valores obtidos a partir da aplicação da heurística lagrangeana e os valores da solução ótima foi em média de $13,86\%$. Em particular, para os problemas 303 e 505, foram obtidos valores de $Gap_{(HLA)/(VO)}$ inferiores a $0,1\%$.

6.5 - Resultados dos Algoritmos GRASP

6.5.1 - Introdução

Nesta seção, temos a apresentação e a análise dos resultados computacionais obtidos a partir da aplicação dos seis algoritmos GRASP propostos para a resolução do PRR e que foram implementados utilizando a linguagem de programação C++.

De acordo com a seção 5.4, estes algoritmos se diferenciam em relação aos procedimentos utilizados na fase de construção e na fase de busca local do GRASP (vide tabela 5.1 - página 49).

Com a utilização destes algoritmos, tivemos a possibilidade de gerar boas soluções viáveis para o PRR de forma razoavelmente rápida (quando comparado com o tempo de processamento da formulação proposta) e utilizar o valor destas soluções viáveis como um limite superior para a nova formulação.

6.5.2 - Análise das Soluções (Desempenho)

Nas tabelas 6.6, 6.7 e 6.8, apresentamos o valor ótimo e o valor da solução obtida a partir da aplicação dos seis algoritmos GRASP (GR1, GR2, GR1F, GR2F, GR1FPR-V0 e GR1FPR-V1) , considerando um conjunto de 70 problemas teste, associados respectivamente aos grafos originais, grafos reduzidos pelo conjunto 1 e grafos reduzidos pelo conjunto 2.

Nestas tabelas, os valores em vermelho indicam que os valores obtidos através dos algoritmos GRASP são iguais ao valor ótimo (considerando os problemas para os quais foi obtida a solução ótima). Para os demais problemas, estes valores indicam a melhor solução obtida através dos algoritmos.

Além da apresentação destes valores, temos o cálculo da diferença percentual (ρ = desempenho) entre a solução ótima e as soluções viáveis dos seis algoritmos, considerando a seguinte expressão:

$$\rho = 100.\frac{sol.viável - sol.ótima}{sol.ótima}$$

A medida de desempenho ρ indica o quanto a solução viável está distante da solução ótima.

Na tabela 6.9, apresentamos o desempenho médio $\bar{\rho}$ de cada um dos seis algoritmos, considerando os 41 problemas teste para os quais se obteve a solução ótima (vide tabelas 6.2 e 6.3). O cálculo deste desempenho médio foi efetuado a partir da seguinte expressão:

$$\bar{\rho} = \frac{\sum \rho}{41}$$

Novamente, para a análise do desempenho ρ e do desempenho médio $\bar{\rho}$, foram considerados os grafos originais e os grafos reduzidos pelo conjunto 1 e pelo conjunto 2.

Com base nestas tabelas, podemos fazer as seguintes observações:

• Para a maioria dos problemas teste, analisando os valores de ρ, os algoritmos GR1FPR-V0 e GR1FPR-V1 forneceram resultados melhores do que os demais algoritmos.

Tal ocorrência pode estar associada a três fatores: Um refinamento das soluções obtidas na fase construção através da aplicação do Filtro, a implementação da metaheurística VNS na fase de busca local e a utilização do procedimento de Reconexão por Caminhos.

• Os algoritmos GR2 e GR2F forneceram soluções piores que os demais algoritmos para a maioria dos problemas teste. Este fato é decorrente da utilização de uma heurística simples na fase de busca local. Nesta heurística, efetuamos apenas a troca simultânea entre dois vértices do conjunto $V \backslash T$.

• Para a maioria dos problemas teste, ao utilizarmos os grafos reduzidos pelo conjunto 2, obtivemos soluções melhores do que aquelas obtidas a partir dos grafos originais e dos grafos reduzidos pelo conjunto 1, independentemente do algoritmo GRASP utilizado.

Este fenômeno pode ser explicado, levando em consideração, que os grafos reduzidos pelo conjunto 2 têm, em sua maioria, um número de vértices menor do que os grafos reduzidos pelo conjunto 1.

• Nos problemas teste com até 60 vértices, foram obtidas soluções iguais ou bem próximas da solução ótima (observando ρ), quando utilizamos os algoritmos GR1FPR-V0 e GR1FPR-V1 (independentemente do tipo de grafo).

• Para os problemas teste com número de vértices maior ou igual a 70 em que não foi possível se obter a solução ótima, considerando o tempo limite de 8 horas, utilizamos a melhor solução viável de um dos seis algoritmos para efetuar o cálculo da diferença percentual ρ .

Em geral, para estes problemas, foram utilizadas as soluções obtidas a partir da utilização dos algoritmos GR1FPR-V0 e GR1FPR-V1, considerando os grafos reduzidos pelo conjunto 2.

• Os resultados da tabela 6.9, indicam que o desempenho médio dos seis algoritmos GRASP foi melhor quando utilizamos os grafos reduzidos pelo conjunto 2. Nesta tabela, os valores destacados em azul indicam qual algoritmo teve o melhor desempenho médio.

Em cada um dos grafos (original, reduzido 1 e reduzido 2) confirmamos o desempenho melhor dos algoritmos GR1FPR-V0 e GR1FPR-V1.

Além das tabelas a seguir, temos no final desta seção, a apresentação de três gráficos onde estão plotadas as soluções viáveis obtidas a partir dos seis algoritmos GRASP e do valor ótimo, considerando apenas os problemas para os quais se obteve a solução ótima.

Estes gráficos reforçam a idéia de que os algoritmos GR2 e GR2F forneceram soluções de qualidade inferior para o PRR, quando comparadas com as soluções dos demais algoritmos.

Tabela 6.6 - Resultados dos Algoritmos Grasp - Grafos Originais

Problemas Teste	Valor Otimo	Valor GR1	ρ	Valor GR2	ρ	Valor GR1F	ρ	Valor GR2F	ρ	Valor GR1FPR-V0	ρ	Valor GR1FPR-V1	ρ
101	1129,3	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%
102	730,3	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%
103	887,6	887,6	0,0%	887,6	0,0%	887,6	0,0%	887,6	0,0%	887,6	0,0%	887,6	0,0%
104	1257,7	1257,7	0,0%	1274,4	1,3%	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%
105	1360,3	1360,3	0,0%	2066,8	51,9%	1360,3	0,0%	1360,3	0,0%	1360,3	0,0%	1360,3	0,0%
201	1920,2	1920,2	0,0%	2493,1	29,8%	1920,2	0,0%	2164,1	12,7%	1920,2	0,0%	1920,2	0,0%
202	1490,8	1490,8	0,0%	1790,9	20,1%	1490,8	0,0%	1839,7	23,4%	1490,8	0,0%	1490,8	0,0%
203	1493,4	1493,4	0,0%	1821,9	22,0%	1493,4	0,0%	1493,4	0,0%	1493,4	0,0%	1493,4	0,0%
204	995,3	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%
205	1721,0	1721,0	0,0%	2622,4	52,4%	1721,0	0,0%	1851,3	7,6%	1721,0	0,0%	1721,0	0,0%
301	1983,1	2052,6	3,5%	2813,8	41,9%	2002,7	1,0%	2196,9	10,8%	1998,2	0,8%	1991,4	0,4%
302	1884,0	2024,4	7,5%	2535,2	34,6%	1959,2	4,0%	2230,4	18,4%	1912,3	1,5%	1903,9	1,1%
303	1966,9	1984,4	0,9%	3858,7	96,2%	1971,9	0,3%	2727,7	38,7%	1981,4	0,7%	1972,5	0,3%
304	1865,6	1897,7	1,7%	2885,5	54,7%	1865,6	0,0%	2143,5	14,9%	1865,6	0,0%	1865,6	0,0%
305	1516,7	1516,7	0,0%	1516,7	0,0%	1517,3	0,0%	2053,4	35,4%	1516,7	0,0%	1516,7	0,0%
401	1691,4	1758,3	4,0%	2566,7	51,8%	1697,7	0,4%	2123,6	25,6%	1691,6	0,0%	1691,4	0,0%
402	2113,8	2126,7	0,6%	3228,1	52,7%	2113,9	0,0%	2875,6	36,0%	2126,7	0,6%	2113,8	0,0%
403	2364,6	2480,7	4,9%	3775,2	59,7%	2367,4	0,1%	3165,6	33,9%	2367,4	0,1%	2367,4	0,1%
404	1987,0	2102,0	5,8%	3875,4	95,0%	2047,9	3,1%	3660,3	84,2%	1987,0	0,0%	1987,0	0,0%
405	1835,3	1961,1	6,9%	3450,0	88,0%	1860,7	1,4%	2643,8	44,1%	1882,7	2,6%	1835,3	0,0%
501	2101,3	2478,3	17,9%	4006,5	90,7%	2306,8	9,8%	2951,0	40,4%	2314,8	10,2%	2225,0	5,9%
502	1915,4	2220,8	15,9%	4460,1	132,9%	2214,9	15,6%	3423,8	78,7%	2114,4	10,4%	2068,2	8,0%
503	2371,4	2453,7	3,5%	3822,2	61,2%	2380,1	0,4%	3029,6	27,8%	2380,1	0,4%	2380,1	0,4%
504	1853,1	1913,2	3,2%	2938,1	58,6%	1854,5	0,1%	2839,6	53,2%	1881,0	1,5%	1874,9	1,2%
505	2171,6	2400,1	10,5%	2849,5	31,2%	2271,3	4,6%	2612,3	20,3%	2279,6	5,0%	2190,5	0,9%
601	2229,3	2820,3	26,5%	3777,8	69,5%	2528,8	13,4%	3016,8	35,3%	2421,7	8,6%	2400,3	7,7%
602	1992,2	2339,6	17,4%	2767,0	38,9%	2101,2	5,5%	2775,7	39,3%	2024,9	1,6%	2025,5	1,7%
603	1678,4	1912,5	13,9%	2000,0	19,2%	1803,0	7,4%	1986,5	18,4%	1698,1	1,2%	1786,3	6,4%
604	1804,3	2419,1	34,1%	2938,9	62,9%	2188,3	21,3%	2658,0	47,3%	2008,5	11,3%	1920,4	6,4%
605	2243,9	2357,4	5,1%	3617,4	61,2%	2266,2	1,0%	3675,0	63,8%	2266,9	1,0%	2280,8	1,6%
701	2335,2	2961,5	26,8%	4646,6	99,0%	2634,7	12,8%	4130,3	76,9%	2636,0	12,9%	2574,6	10,3%
702	2537,2	2810,0	10,7%	5429,9	114,0%	2748,1	8,3%	3424,2	35,0%	2723,1	7,3%	2723,1	7,3%
703	2629,0	2904,3	10,5%	5377,2	104,5%	2839,6	8,0%	4737,1	80,2%	2754,5	4,8%	2791,7	6,2%
704*	2883,7	3029,7	5,1%	3619,3	25,5%	2943,3	2,1%	3332,3	15,6%	2909,3	0,9%	2926,1	1,5%
705	1702,1	2045,0	20,1%	2888,3	69,7%	1981,4	16,4%	2477,4	45,6%	1907,8	12,1%	1781,4	4,7%
801	2303,9	2966,8	28,8%	4553,4	97,6%	2750,4	19,4%	3671,8	59,4%	2643,6	14,7%	2590,6	12,4%
802*	2771,5	3116,5	12,4%	4398,7	58,7%	2966,4	7,0%	4036,5	45,6%	2855,6	3,0%	2836,9	2,4%
803*	2947,8	3215,6	9,1%	3613,6	22,6%	3155,2	7,0%	3192,0	8,3%	3153,9	7,0%	3170,8	7,6%
804	2312,8	2915,3	26,0%	3317,5	43,4%	2709,9	17,2%	3013,4	30,3%	2700,5	16,8%	2596,4	12,3%
805	2343,5	2960,9	26,3%	3863,5	64,9%	2671,2	14,0%	3790,2	61,7%	2647,4	13,0%	2647,3	13,0%
901*	2666,9	3132,6	17,5%	4566,4	71,2%	3036,0	13,8%	4517,9	69,4%	2818,3	5,7%	2955,8	10,8%
902*	3162,4	3714,6	17,5%	5520,6	74,6%	3482,2	10,1%	5190,9	64,1%	3405,1	7,7%	3427,1	8,4%
903*	2904,9	3444,2	18,6%	6892,9	137,3%	3082,1	6,1%	5984,7	106,0%	3039,3	4,6%	3116,6	7,3%
904*	2256,4	2941,1	30,3%	3347,9	48,4%	2759,8	22,3%	2698,9	19,6%	2593,6	14,9%	3246,9	43,9%
905*	3180,9	3654,6	14,9%	4595,9	44,5%	3382,1	6,3%	4191,5	31,8%	3282,7	3,2%	3299,1	3,7%
1001*	2610,2	3564,7	36,6%	3790,0	45,2%	2622,9	0,5%	3710,6	42,2%	3545,9	35,8%	3313,2	26,9%
1002	2308,8	2762,4	19,6%	3761,8	62,9%	2308,8	0,0%	3333,2	44,4%	2660,5	15,2%	2442,8	5,8%
1003*	2832,7	3268,9	15,4%	3545,4	25,2%	2832,7	0,0%	3091,8	9,1%	3112,6	9,9%	3039,1	7,3%
1004	2604,1	2989,9	14,8%	4680,3	79,7%	2736,6	5,1%	4052,1	55,6%	2747,2	5,5%	2791,5	7,2%
1005	2496,3	2944,5	18,0%	2948,9	18,1%	2501,1	0,2%	2501,1	0,2%	2739,6	9,7%	2666,6	6,8%
2001*	3556,7	5309,8	49,3%	5202,8	46,3%	3617,9	1,7%	3991,0	12,2%	5070,4	42,6%	4884,5	37,3%
2002	2803,8	2979,1	6,3%	5341,4	90,5%	2861,3	2,0%	4619,4	64,8%	2955,7	5,4%	2844,6	1,5%
2003*	3752,9	5054,1	34,7%	9165,6	144,2%	4010,5	6,9%	8491,3	126,3%	4662,6	24,2%	4616,4	23,0%
2004*	3512,8	4742,5	35,0%	10659,0	203,4%	3720,5	5,9%	7912,9	125,3%	4628,6	31,8%	4340,9	23,6%
2005*	4393,6	5349,5	21,8%	5359,9	22,0%	4468,1	1,7%	4735,2	7,8%	5174,3	17,8%	5064,0	15,3%
3001*	4945,3	6665,5	34,8%	8529,7	72,5%	6498,3	31,4%	6949,9	40,5%	6520,8	31,9%	6450,7	30,4%
3002*	4179,7	5421,1	29,7%	10458,5	150,2%	5421,1	29,7%	9059,6	116,8%	5060,1	21,1%	4946,1	18,3%
3003*	5503,8	6597,4	19,9%	8602,2	56,3%	6597,4	19,9%	7944,0	44,3%	6485,9	17,8%	6414,7	16,6%
3004*	5671,7	6624,2	16,8%	8814,4	55,4%	6674,8	17,7%	8059,2	42,1%	6371,1	12,3%	6346,4	11,9%
3005*	4638,7	5385,5	16,1%	11295,3	143,5%	5519,8	19,0%	9652,2	108,1%	5300,8	14,3%	5183,4	11,7%
4001*	6342,7	7889,0	24,4%	14782,8	133,1%	7947,0	25,3%	11372,0	79,3%	7765,5	22,4%	7776,7	22,6%
4002*	5332,0	7695,8	44,3%	7902,7	48,2%	7752,7	45,4%	7060,6	32,4%	7568,8	42,0%	7289,4	36,7%
4003*	6484,5	7736,4	19,3%	11766,8	81,5%	7736,4	19,3%	10435,0	60,9%	7518,9	16,0%	7519,5	16,0%
4004*	5078,1	6708,7	32,1%	10794,0	112,6%	6756,8	33,1%	8585,0	69,1%	6504,3	28,1%	6463,8	27,3%
4005*	4963,1	7659,1	54,3%	9154,2	84,4%	7507,2	51,3%	7623,0	53,6%	7492,5	51,0%	7175,0	44,6%
5001*	6667,1	8716,7	30,7%	12825,8	92,4%	8488,3	27,3%	12767,0	91,5%	8578,0	28,7%	8472,3	27,1%
5002*	5496,9	8561,2	55,7%	13042,1	137,3%	8475,1	54,2%	9298,5	69,2%	8321,8	51,4%	8317,9	51,3%
5003*	8126,1	9310,7	14,6%	10420,8	28,2%	9216,8	13,4%	9607,6	18,2%	9105,4	12,1%	9244,5	13,8%
5004*	6637,3	8880,9	33,8%	7800,3	17,5%	8710,4	31,2%	7347,0	10,7%	8741,3	31,7%	8741,3	31,7%
5005*	5593,5	8211,9	46,8%	11881,4	112,4%	8080,7	44,5%	11288,4	101,8%	8080,1	44,5%	8368,8	49,6%

***Foi utilizada a melhor solução viável**

Tabela 6.7 - Resultados dos Algoritmos Grasp - Grafos Reduzidos pelo Conjunto 1

Problemas Teste	Valor Otimo	Valor GR1	ρ	Valor GR2	ρ	Valor GR1F	ρ	Valor GR2F	ρ	Valor GR1FPR-V0	ρ	Valor GR1FPR-V1	ρ
101	1129,3	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%
102	730,3	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%
103	887,6	887,6	0,0%	887,6	0,0%	887,6	0,0%	887,6	0,0%	887,6	0,0%	887,6	0,0%
104	1257,7	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%
105	1360,3	1360,3	0,0%	2066,8	51,9%	1360,3	0,0%	1360,3	0,0%	1360,3	0,0%	1360,3	0,0%
201	1920,2	1920,2	0,0%	2511,6	30,8%	1920,2	0,0%	2481,3	29,2%	1920,2	0,0%	1920,2	0,0%
202	1490,8	1490,8	0,0%	1790,9	20,1%	1490,8	0,0%	1790,9	20,1%	1490,8	0,0%	1490,8	0,0%
203	1493,4	1493,4	0,0%	2353,2	57,6%	1493,4	0,0%	1584,4	6,1%	1493,4	0,0%	1493,4	0,0%
204	995,3	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%
205	1721,0	1721,0	0,0%	2949,3	71,4%	1721,0	0,0%	2071,1	20,3%	1721,0	0,0%	1721,0	0,0%
301	1983,1	1983,1	0,0%	2318,4	16,9%	1983,1	0,0%	2277,9	14,9%	1983,1	0,0%	1983,1	0,0%
302	1884,0	1884,0	0,0%	1944,8	3,2%	1884,0	0,0%	1888,1	0,2%	1884,0	0,0%	1884,0	0,0%
303	1966,9	1969,3	0,1%	3503,5	78,1%	1971,9	0,3%	3166,6	61,0%	1966,9	0,0%	1966,9	0,0%
304	1865,6	1865,6	0,0%	2885,5	54,7%	1865,6	0,0%	2269,1	21,6%	1865,6	0,0%	1865,6	0,0%
305	1516,7	1516,7	0,0%	2175,6	43,4%	1516,7	0,0%	2053,4	35,4%	1516,7	0,0%	1694,6	11,7%
401	1691,4	1691,4	0,0%	2688,2	58,9%	1691,4	0,0%	2511,3	48,5%	1691,4	0,0%	1691,4	0,0%
402	2113,8	2113,9	0,0%	3163,2	49,6%	2113,8	0,0%	2798,0	32,4%	2113,8	0,0%	2113,8	0,0%
403	2364,6	2368,2	0,2%	3598,4	52,2%	2364,6	0,0%	3062,4	29,5%	2364,6	0,0%	2364,6	0,0%
404	1987,0	1987,0	0,0%	3732,8	87,9%	1987,0	0,0%	3665,5	84,5%	1987,0	0,0%	1987,0	0,0%
405	1835,3	1850.7	0,8%	3813.6	107,8%	1850.7	0,8%	2372.9	29.3%	1835.3	0.0%	1835.3	0.0%
501	2101,3	2286,8	8,8%	4871,2	131,8%	2141,0	1,9%	3693,6	75,8%	2102,2	0,0%	2107,3	0,3%
502	1915,4	2011,8	5,0%	4679,5	144,3%	2011,8	5,0%	3372,8	76,1%	2003,1	4,6%	1997,8	4,3%
503	2371,4	2371.4	0,0%	3768.5	58,9%	2371.4	0.0%	3260.7	37.5%	2371.4	0.0%	2371.4	0.0%
504	1853,1	1854.5	0,1%	3582.8	93,3%	1854.5	0.1%	3204.7	72.9%	1882.7	1.6%	1853.1	0.0%
505	2171,6	2171.6	0,0%	3739.6	72,2%	2171.6	0.0%	3171.2	46.0%	2171.6	0.0%	2171.6	0.0%
601	2229,3	2358.0	5,8%	3780.8	69,6%	2305.4	3.4%	3621.8	62.5%	2246.0	0.7%	2262.0	1.5%
602	1992,2	2094.3	5,1%	2778.2	39,5%	1992.2	0.0%	2464.4	23.7%	1992.2	0.0%	1992.2	0.0%
603	1678,4	1893.2	12,8%	1970.4	17,4%	1900.1	13.2%	1900.1	13.2%	1678.4	0.0%	1678.4	0.0%
604	1804,3	1998.7	10,8%	3093.9	71,5%	1946.5	7.9%	2871.3	59.1%	1851.6	2.6%	1876.1	4.0%
605	2243,9	2348.9	4,7%	3897.7	73,7%	2251.9	0.4%	3512.3	56.5%	2243.9	0.0%	2243.9	0.0%
701	2335,2	2743.0	17,5%	4451.3	90,6%	2656.0	13.7%	3963.5	69.7%	2578.1	10.4%	2619.1	12.2%
702	2537,2	2563.9	1,1%	3242.8	27,8%	2557.4	0.8%	2799.9	10.4%	2558.4	0.8%	2557.4	0.8%
703	2629,0	2700.5	2,7%	4908.2	86,7%	2640.2	0.4%	4148.1	57.8%	2651.4	0.9%	2632.9	0.1%
704*	2883,7	2940,1	2,0%	3797,2	31,7%	2910,0	0,9%	3291,4	14,1%	2904,8	0,7%	2905,8	0,8%
705	1702,1	1794,1	5,4%	3716,6	118,4%	1794,1	5,4%	2537,2	49,1%	1702,1	0,0%	1758,1	3,3%
801	2303,9	2422.3	5,1%	3550.1	54,1%	2326.0	1.0%	3047.9	32.3%	2350.7	2.0%	2335.5	1.4%
802*	2771,5	2995,0	8,1%	4689,8	69,2%	2814,1	1,5%	4050,1	46,1%	2771,5	0,0%	2778,8	0,3%
803*	2947,8	3117,7	5,8%	3027,2	2,7%	2949,8	0,1%	2976,1	1,0%	2947,8	0,0%	2976,1	1,0%
804	2312,8	2545,3	10,0%	3934,7	70,1%	2443,8	5,7%	3481,9	50,5%	2387,6	3,2%	2392,4	3,4%
805	2343,5	2888,5	23,3%	3659,6	56,2%	2478,9	5,8%	3557,3	51,8%	2438,9	4,1%	2447,7	4,4%
901*	2666,9	2860,1	7,2%	5113,5	91,7%	2753,0	3,2%	5240,4	96,5%	2671,0	0,2%	2702,9	1,4%
902*	3162,4	3533,0	11,7%	4814,5	52,2%	3229,8	2,1%	4633,4	46,5%	3245,2	2,6%	3162,4	0,0%
903*	2904,9	3108,2	7,0%	6661,7	129,3%	2988,5	2,9%	6334,2	118,0%	2946,6	1,4%	2904,9	0,0%
904*	2256,4	2584,6	14,5%	3401,0	50,7%	2395,4	6,2%	3260,1	44,5%	2256,4	0,0%	2289,7	1,5%
905*	3180,9	3425,8	7,7%	4118,8	29,5%	3250,1	2,2%	3825,4	20,3%	3248,3	2,1%	3252,1	2,2%
1001*	2610,2	2645,5	1,4%	3717,6	42,4%	2622,9	0,5%	3215,8	23,2%	2610,2	0,0%	2610,2	0,0%
1002	2308,8	2308,8	0,0%	3063,0	32,7%	2308,8	0,0%	2903,8	25,8%	2308,8	0,0%	2308,8	0,0%
1003*	2832,7	2888,9	2,0%	3618,7	27,7%	2832,7	0,0%	3019,2	6,6%	2877,9	1,6%	2863,8	1,1%
1004	2604,1	2885,4	10,8%	5889,6	126,2%	2736,6	5,1%	5377,2	106,5%	2719,3	4,4%	2668,8	2,5%
1005	2496,3	2523,6	1,1%	2929,5	17,4%	2501,1	0,2%	2567,1	2,8%	2501,1	0,2%	2500,4	0,2%
2001*	3556,7	3677,0	3,4%	4935,1	38,8%	3617,9	1,7%	4421,6	24,3%	3586,8	0,8%	3603,9	1,3%
2002	2803,8	2923,8	4,3%	6208,8	121,4%	2861,3	2,0%	5276,9	88,2%	2846,3	1,5%	2836,0	1,1%
2003*	3752,9	4071,7	8,5%	12758,2	240,0%	4010,5	6,9%	10025,6	167,1%	3878,1	3,3%	3784,8	0,8%
2004*	3512,8	3937.8	12,1%	11372,3	223,7%	3720.5	5.9%	8662.5	146.6%	3682.6	4.8%	3691.1	5.1%
2005*	4393,6	4659.5	6,1%	5476.0	24,6%	4468.1	1.7%	4626.2	5.3%	4427.3	0.8%	4428.4	0.8%
3001*	4945,3	5182,6	4,8%	8219,0	66,2%	5182,6	4,8%	5914,1	19,6%	4966,2	0,4%	5010,7	1,3%
3002*	4179,7	4610,7	10,3%	11281,8	169,9%	4610,7	10,3%	9810,1	134,7%	4356,2	4,2%	4387,8	5,0%
3003*	5503,8	5872.9	6,7%	8925.9	62,2%	5872.9	6.7%	8109.4	47.3%	5503.8	0.0%	5706.8	3.7%
3004*	5671,7	5940.8	4,7%	8805.3	55,2%	5940.8	4.7%	7550.6	33.1%	5719.0	0.8%	5687.2	0.3%
3005*	4638,7	4994.8	7,7%	11552.3	149,0%	4994.8	7.7%	7656.8	65.1%	4776.5	3.0%	4762.9	2.7%
4001*	6342,7	6711.8	5,8%	11992,1	89,1%	6711.8	5.8%	10575.5	66.7%	6410.4	1.1%	6342.7	0.0%
4002*	5332,0	5481.4	2,8%	6678.3	25,2%	5492.2	3.0%	6151.4	15.4%	5397.0	1.2%	5332.0	0.0%
4003*	6484,5	6841.5	5,5%	10799.4	66,5%	6882.0	6.1%	9973.2	53.8%	6639.1	2.4%	6639.1	2.4%
4004*	5078,1	5462.5	7,6%	8634.3	70,0%	5462.5	7.6%	7507.7	47.8%	5274.9	3.9%	5158.8	1.6%
4005*	4963,1	5393.4	8,7%	10094.2	103,4%	5208.2	4.9%	8210.3	65.4%	5074.5	2.2%	5032.7	1.4%
5001*	6667,1	6973.0	4,6%	12272.7	84,1%	6738.5	1.1%	11737.3	76.0%	6667.1	0.0%	6677.9	0.2%
5002*	5496,9	6007.5	9,3%	10945.1	99,1%	5821.4	5.9%	9880.9	79.8%	5826.2	6.0%	5496.9	0.0%
5003*	8126,1	8388.4	3,2%	10076.9	24,0%	8315.7	2.3%	9842.5	21.1%	8218.1	1.1%	8293.6	2.1%
5004*	6637,3	6887,3	3,8%	8660,0	30,5%	6637,3	0,0%	8394,2	26,5%	6676,0	0,6%	6720,3	1,2%
5005*	5593,5	6054.8	8,2%	14326.2	156,1%	5881.6	5.2%	12211.3	118.3%	5856.2	4.7%	5821.2	4.1%

***Foi utilizada a melhor solução viável**

Tabela 6.8 - Resultados dos Algoritmos Grasp - Grafos Reduzidos pelo Conjunto 2

Problemas Teste	Valor Otimo	Valor GR1	ρ	Valor GR2	ρ	Valor GR1F	ρ	Valor GR2F	ρ	Valor GR1FPR-V0	ρ	Valor GR1FPR-V1	ρ
101	1129,3	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%	1129,3	0,0%
102	730,3	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%	730,3	0,0%
103	887,6	887,6	0,0%	1032,8	16,4%	887,6	0,0%	887,6	0,0%	887,6	0,0%	887,6	0,0%
104	1257,7	1257,7	0,0%	1274,4	1,3%	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%	1257,7	0,0%
105	1360,3	1360,3	0,0%	1413,1	3,9%	1360,3	0,0%	1360,3	0,0%	1360,3	0,0%	1360,3	0,0%
201	1920,2	1920,2	0,0%	1920,2	0,0%	1920,2	0,0%	1920,2	0,0%	1920,2	0,0%	1920,2	0,0%
202	1490,8	1490,8	0,0%	1994,7	33,8%	1490,8	0,0%	1790,9	20,1%	1490,8	0,0%	1490,8	0,0%
203	1493,4	1493,4	0,0%	1711,6	14,6%	1493,4	0,0%	1493,4	0,0%	1493,4	0,0%	1493,4	0,0%
204	995,3	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%	995,3	0,0%
205	1721,0	1721,0	0,0%	1980,0	15,0%	1721,0	0,0%	1721,0	0,0%	1721,0	0,0%	1721,0	0,0%
301	1983,1	1983,1	0,0%	2158,5	8,8%	1983,1	0,0%	1983,1	0,0%	1983,1	0,0%	1983,1	0,0%
302	1884,0	1884,0	0,0%	2170,3	15,2%	1884,0	0,0%	1884,0	0,0%	1884,0	0,0%	1884,0	0,0%
303	1966,9	1966,9	0,0%	3404,3	73,1%	1966,9	0,0%	2592,0	31,8%	1966,9	0,0%	1971,9	0,3%
304	1865,6	1865,6	0,0%	2426,5	30,1%	1865,6	0,0%	1920,1	2,9%	1865,6	0,0%	1865,6	0,0%
305	1516,7	1516,7	0,0%	2072,1	36,6%	1516,7	0,0%	1809,4	19,3%	1516,7	0,0%	1516,7	0,0%
401	1691,4	1691,4	0,0%	2270,4	34,2%	1691,4	0,0%	1730,7	2,3%	1691,4	0,0%	1691,4	0,0%
402	2113,8	2113,9	0,0%	2478,6	17,3%	2113,8	0,0%	2199,7	4,1%	2113,8	0,0%	2113,8	0,0%
403	2364,6	2364,6	0,0%	3441,5	45,5%	2364,6	0,0%	2968,0	25,5%	2364,6	0,0%	2364,6	0,0%
404	1987,0	1987,0	0,0%	2320,0	16,8%	1987,0	0,0%	2061,3	3,7%	1987,0	0,0%	1987,0	0,0%
405	1835,3	1835,3	0,0%	2431,0	32,5%	1835,3	0,0%	2109,9	15,0%	1835,3	0,0%	1835,3	0,0%
501	2101,3	2101,3	0,0%	2420,6	15,2%	2101,3	0,0%	2122,3	1,0%	2101,3	0,0%	2101,3	0,0%
502	1915,4	1970,5	2,9%	3643,0	90,2%	1970,5	2,9%	2974,5	55,3%	1978,8	3,3%	2009,3	4,9%
503	2371,4	2414,4	1,8%	3413,4	43,9%	2371,4	0,0%	3256,1	37,3%	2371,4	0,0%	2371,4	0,0%
504	1853,1	1886,9	1,8%	2955,7	59,5%	1886,9	1,8%	2592,0	39,9%	1875,0	1,2%	1853,1	0,0%
505	2171,6	2171,6	0,0%	3162,2	45,6%	2171,6	0,0%	2501,3	15,2%	2171,6	0,0%	2171,6	0,0%
601	2229,3	2333,3	4,7%	3562,1	59,8%	2289,7	2,7%	2867,3	28,6%	2246,0	0,7%	2255,2	1,2%
602	1992,2	2001,3	0,5%	2556,3	28,3%	1994,5	0,1%	2293,1	15,1%	1992,2	0,0%	1995,0	0,1%
603	1678,4	1893,2	12,8%	1886,4	12,4%	1762,8	5,0%	1886,4	12,4%	1678,4	0,0%	1678,4	0,0%
604	1804,3	2013,5	11,6%	2654,7	47,1%	1949,5	8,0%	2148,3	19,1%	1893,6	4,9%	1868,2	3,5%
605	2243,9	2342,1	4,4%	3077,4	37,1%	2243,9	0,0%	3033,2	35,2%	2246,8	0,1%	2243,9	0,0%
701	2335,2	2776,2	18,9%	5110,4	118,8%	2725,2	16,7%	3635,7	55,7%	2547,2	9,1%	2479,6	6,2%
702	2537,2	2576,8	1,6%	2936,0	15,7%	2537,2	0,0%	2645,2	4,3%	2538,2	0,0%	2563,9	1,1%
703	2629,0	2717,7	3,4%	3704,2	40,9%	2636,3	0,3%	3324,7	26,5%	2644,2	0,6%	2676,3	1,8%
704*	2883,7	2937,0	1,9%	3563,4	23,6%	2913,5	1,0%	2974,6	3,2%	2883,7	0,0%	2914,4	1,1%
705	1702,1	1832,8	7,7%	2932,6	72,3%	1813,2	6,5%	2501,7	47,0%	1762,2	3,5%	1766,4	3,8%
801	2303,9	2536,4	10,1%	3073,1	33,4%	2366,4	2,7%	2838,6	23,2%	2354,1	2,2%	2366,4	2,7%
802*	2771,5	3165,5	14,2%	3482,8	25,7%	2796,0	0,9%	3343,9	20,7%	2789,2	0,6%	2788,3	0,6%
803*	2947,8	3110,7	5,5%	3276,5	11,2%	2947,8	0,0%	3151,8	6,9%	2977,2	1,0%	2947,8	0,0%
804	2312,8	2382,3	3,0%	3732,0	61,4%	2326,3	0,6%	2593,8	12,1%	2326,3	0,6%	2316,6	0,2%
805	2343,5	2445,2	4,3%	2582,6	10,2%	2343,5	0,0%	2428,7	3,6%	2343,5	0,0%	2343,5	0,0%
901*	2666,9	3008,2	12,8%	3783,3	41,9%	2718,3	1,9%	3339,7	25,2%	2666,9	0,0%	2681,6	0,6%
902*	3162,4	3571,4	12,9%	4367,4	38,1%	3240,9	2,5%	3811,9	20,5%	3207,6	1,4%	3236,6	2,3%
903*	2904,9	3072,8	5,8%	5104,9	75,7%	2988,5	2,9%	4472,3	54,0%	2925,3	0,7%	2966,0	2,1%
904*	2256,4	2580,6	14,4%	3379,8	49,8%	2373,4	5,2%	2808,2	24,5%	2283,6	1,2%	2304,6	2,1%
905*	3180,9	3529,1	10,9%	4196,3	31,9%	3231,6	1,6%	3591,1	12,9%	3180,9	0,0%	3201,0	0,6%
1001*	2610,2	2630,2	0,8%	3097,0	18,6%	2610,2	0,0%	2634,5	0,9%	2610,2	0,0%	2621,7	0,4%
1002	2308,8	2308,8	0,0%	2813,3	21,9%	2308,8	0,0%	2455,9	6,4%	2308,8	0,0%	2308,8	0,0%
1003*	2832,7	2982,9	5,3%	3603,7	27,2%	2842,4	0,3%	2986,4	5,4%	2840,1	0,3%	2864,2	1,1%
1004	2604,1	2834,6	8,9%	4265,0	63,8%	2720,4	4,5%	3479,4	33,6%	2632,6	1,1%	2701,1	3,7%
1005	2496,3	2548,7	2,1%	2548,7	2,1%	2506,4	0,4%	2585,5	3,6%	2501,8	0,2%	2506,7	0,4%
2001*	3556,7	3816,1	7,3%	5015,6	41,0%	3556,7	0,0%	4322,2	21,5%	3606,2	1,4%	3609,1	1,5%
2002	2803,8	2915,8	4,0%	3418,3	21,9%	2814,4	0,4%	2967,1	5,8%	2829,0	0,9%	2821,4	0,6%
2003*	3752,9	4131,6	10,1%	11423,5	204,4%	3946,8	5,2%	8923,0	137,8%	3818,8	1,8%	3752,9	0,0%
2004*	3512,8	3953,1	12,5%	8904,8	153,5%	3729,9	6,2%	7074,6	101,4%	3545,4	0,9%	3512,8	0,0%
2005*	4393,6	4547,2	3,5%	5735,3	30,5%	4472,9	1,8%	4813,2	9,6%	4393,6	0,0%	4466,1	1,6%
3001*	4945,3	5111,1	3,4%	7693,8	55,6%	5111,1	3,4%	5880,7	18,9%	4945,3	0,0%	5006,8	1,2%
3002*	4179,7	4445,4	6,4%	12082,8	189,1%	4445,4	6,4%	9334,4	123,3%	4278,6	2,4%	4179,7	0,0%
3003*	5503,8	5858,1	6,4%	7246,5	31,7%	5969,3	8,5%	6579,5	19,5%	5650,1	2,7%	5565,2	1,1%
3004*	5671,7	5753,0	1,4%	7787,8	37,3%	5753,0	1,4%	6643,9	17,1%	5776,9	1,9%	5671,7	0,0%
3005*	4638,7	4849,9	4,6%	7403,8	59,6%	4849,9	4,6%	5351,1	15,4%	4680,3	0,9%	4638,7	0,0%
4001*	6342,7	6596,3	4,0%	11142,6	75,7%	6596,3	4,0%	8921,3	40,7%	6367,7	0,4%	6354,1	0,2%
4002*	5332,0	5548,8	4,1%	7047,1	32,2%	5596,1	5,0%	6176,0	15,8%	5388,8	1,1%	5389,7	1,1%
4003*	6484,5	6872,7	6,0%	9572,6	47,6%	6872,7	6,0%	8635,3	33,2%	6484,5	0,0%	6624,0	2,2%
4004*	5078,1	5404,2	6,4%	8812,4	73,5%	5404,2	6,4%	7622,2	50,1%	5149,3	1,4%	5078,1	0,0%
4005*	4963,1	5293,9	6,7%	7604,7	53,2%	5173,7	4,2%	6822,5	37,5%	5073,9	2,2%	4963,1	0,0%
5001*	6667,1	6977,6	4,7%	11387,9	70,8%	6714,8	0,7%	11326,9	69,9%	6690,4	0,3%	6677,0	0,1%
5002*	5496,9	6005,5	9,3%	9938,3	80,8%	5671,8	3,2%	9041,1	64,5%	5726,2	4,2%	5582,2	1,6%
5003*	8126,1	8404,9	3,4%	9464,0	16,5%	8231,9	1,3%	9238,1	13,7%	8225,3	1,2%	8126,1	0,0%
5004*	6637,3	6954,7	4,8%	8692,2	31,0%	6693,5	0,8%	8518,9	28,3%	6695,6	0,9%	6709,1	1,1%
5005*	5593,5	5903,0	5,5%	9414,5	68,3%	5721,3	2,3%	8521,8	52,4%	5593,5	0,0%	5644,9	0,9%

***Foi utilizada a melhor solução viável**

Tabela 6.9 - Desempenho Médio dos Algoritmos GRASI

Grafo	GR1	GR2	GR1F	GR2F	GR1FPR-V0	GR1FPR-V1
Original	9,6%	54,2%	4,7%	33,3%	4,3%	**3,2%**
Reduzido Conj. 1	3,3%	57,5%	1,8%	36,6%	**0,9%**	1,2%
Reduzido Conj. 2	2,5%	31,6%	1,3%	14,8%	**0,7%**	**0,7%**

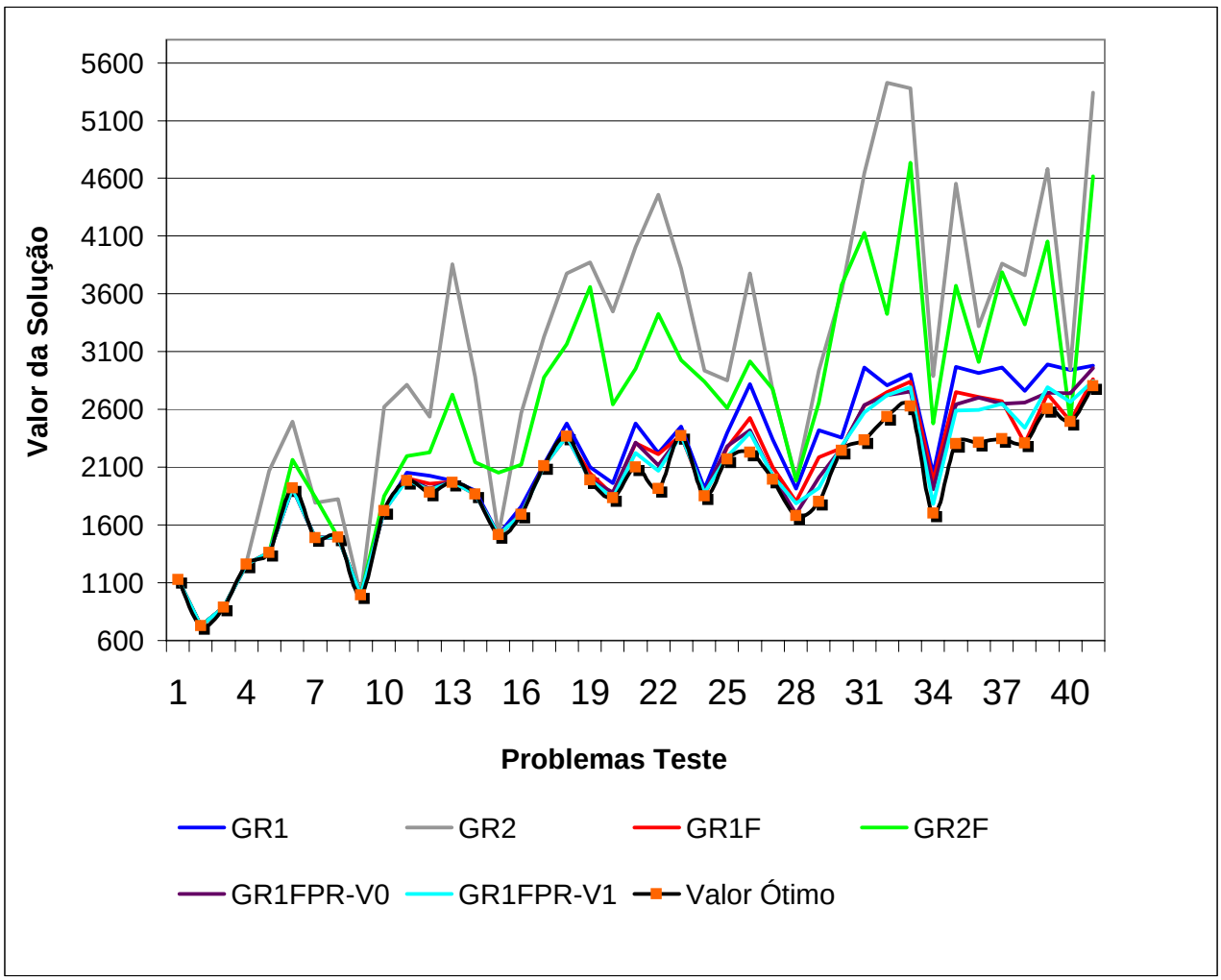

Gráfico 6.3 - Análise Gráfica dos Resultados dos Algoritmos GRASP (grafos originais)

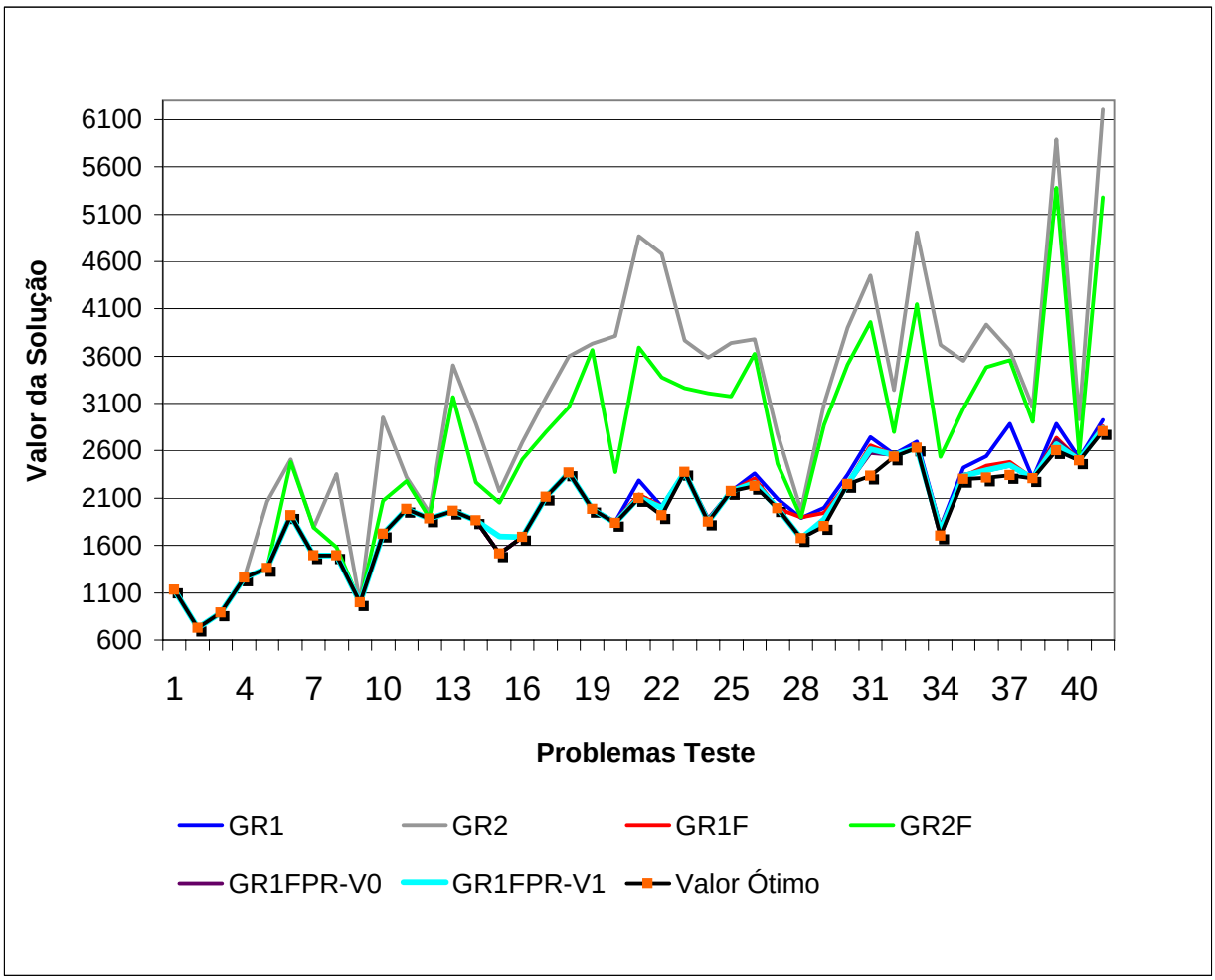

Gráfico 6.4 - Análise Gráfica dos Resultados dos Algoritmos GRASP (grafos reduzidos pelo conj. 1)

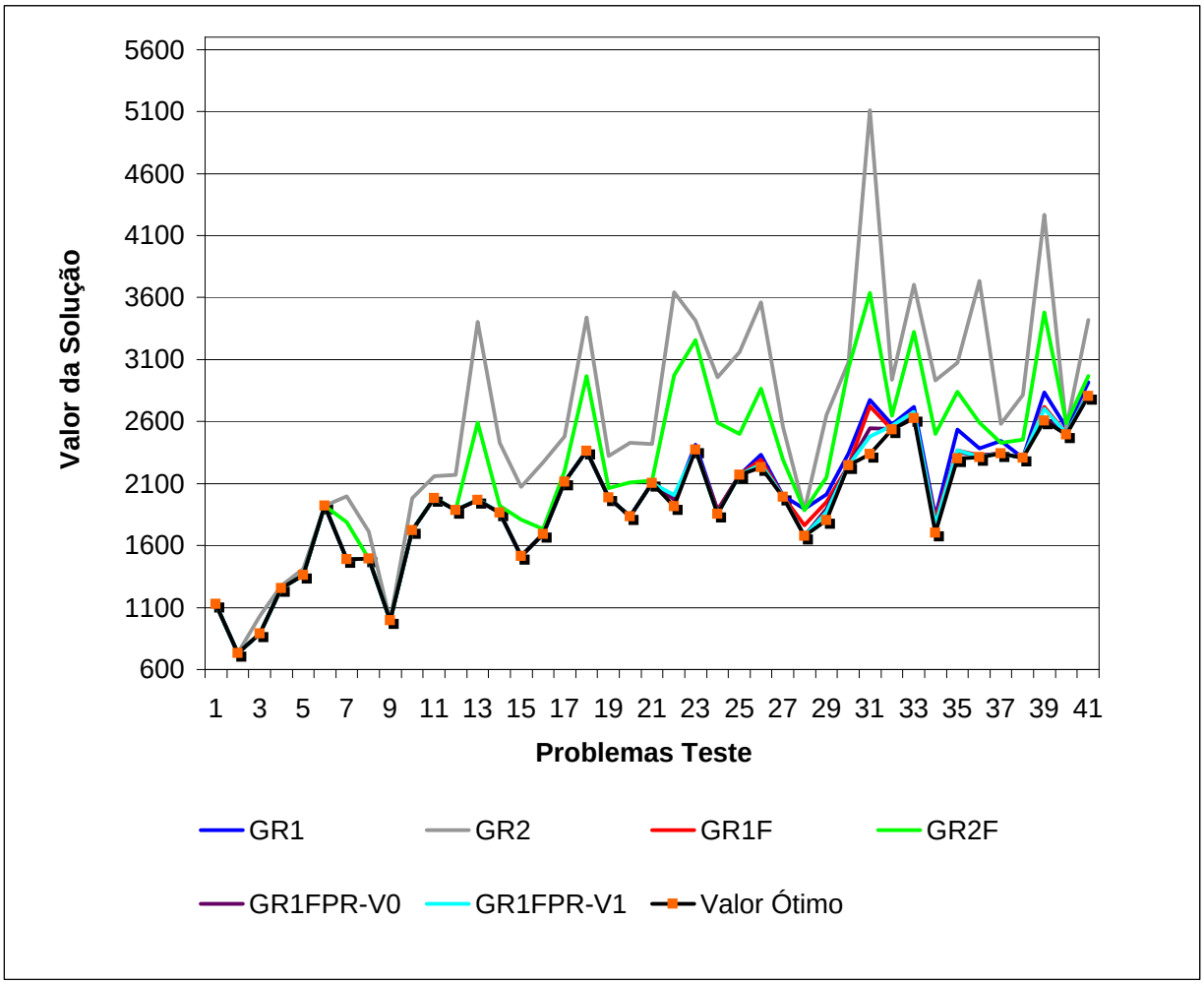

Gráfico 6.5 - Análise Gráfica dos Resultados dos Algoritmos GRASP (grafos reduzidos pelo conj. 2)

6.5.3 - Análise Probabilística

Para avaliar o desempenho dos algoritmos GRASP propostos neste trabalho, efetuamos a análise probabilística (Rezende[34,37]) considerando dois problemas teste.

Para cada um destes problemas, executamos os seis algoritmos GRASP 200 vezes e calculamos a solução média, obtida a partir da média das soluções observadas nas 200 iterações dos algoritmos. E desta forma, obtivemos seis soluções médias associadas a cada um dos problemas.

A partir destas soluções foram definidos três tipos alvos, utilizados para análise probabilística: Alvo Fácil dado pela pior solução média de um dos algoritmos; Alvo Médio, dado pela média das soluções médias dos seis algoritmos; e finalmente Alvo Difícil, dado pela melhor solução média de um dos algoritmos.

Na tabela 6.10, temos os alvos definidos para os problemas 1001 (100 vértices) e 2002 (200 vértices), considerando o grafo original e os grafos reduzidos (R_1 *e* R_2).

Tabela 6.10 - Alvos dos Problemas 1001 e 2002

	Problema 1001			Problema 2002		
Alvo	Orig.	R1	R2	Orig.	R1	R2
Fácil	3858	3365	2971	4826	5469	3137
Médio	3590	2851	2729	3570	3692	2938
Difícil	3396	2611	2612	2939	2830	2814

Após a determinação destes três alvos, a análise foi feita da seguinte maneira: Cada algoritmo foi executado 100 vezes para cada problema teste. No instante em que o algoritmo encontrava uma solução menor ou igual ao alvo, o tempo de processamento era inserido em uma lista L e uma nova execução começava. Ou seja, um novo critério de parada foi inserido para cada algoritmo.

Concluídas as 100 iterações do algoritmo, ordenamos os tempos armazenados em L em ordem crescente. Para cada tempo de processamento t_i obtido associamos a probabilidade $p_i = (i - \frac{1}{2}) / 100$. Em seguida, os pontos $z_i = (t_i, p_i)$ foram plotados em um gráfico, estabelecendo uma distribuição de probabilidade empírica do tempo que cada algoritmo consome para alcançar ao alvo pré-determinado.

Nos casos em que o algoritmo não atinge o alvo nas 100 execuções consecutivas, admitimos que o mesmo é incapaz de atingí-lo, o que o faz ficar fora da análise para tal alvo.

Nos gráficos (6.6 a 6.23) utilizados para a análise probabilística dos dois problemas teste, temos plotados os tempo processamento dos seis algoritmos e as suas probabilidades de atingir os alvos. Observamos, que nestes gráficos, quanto mais à esquerda estiver a curva associada a um algoritmo, melhor será o seu desempenho.

Problema 1001 - Grafo Original

A probabilidade de que os algoritmos GR1,GR2,GR2F e GR1FPR-V0 atinjam o alvo fácil em torno de 100 milisegundos é de aproximadamente 93% , subindo para 100% no caso do algoritmo GR1F. E para o algoritmo GR1FPR-V1, esta probabilidade fica próxima de 40% .

No caso do alvo fácil, o algoritmo GR2 é o que apresentou melhor desempenho, considerando que a probabilidade de encontrar este alvo no tempo de 10 milisegundos é de aproximadamente 90% .

Esta análise ressalta que no caso do alvo fácil, não há necessidade de utilizar os algoritmos GRASP que têm o procedimentos de Filtro e de Reconexão por Caminhos (GR1F, GR2F, GR1FPR-V0 e GR1FPR-V1), pois estes procedimentos tendem a consumir um tempo maior de processamento.

Podemos observar, que para um tempo em torno de 100 milisegundos, a probabilidade de que o algoritmo GR1FPR-V1 encontre o alvo médio é de quase 60% , aumentando para 70% para o algoritmo GR1FPR-V0 e quase 90% para o algoritmo GR1F.

Nesta mesma análise vimos a dificuldade dos algoritmos GR1, GR2 e GR2F em atingirem o alvo médio, em sua totalidade. O algoritmo GR2 só atingiu este alvo apenas uma vez (considerando as 100 execuções) no tempo de 3775 milisegundos. Enquanto os algoritmos GR1 e GR2F atingiram este alvo em um pouco mais de 80% dos casos.

Além disso, quando consideramos um tempo superior a 10000 milisegundos, apenas os algoritmos GR1FPR-V0 e GR1FPR-V1 têm probabilidade de 100% de atingir o alvo médio.

Desta forma, à medida que reduzimos o valor do alvo (de fácil para médio), torna-se vantajoso utilizar os algoritmos GRASP com a metaheurística VNS e os procedimentos de Filtro e de Reconexão por Caminhos.

Quando consideramos o alvo difícil, apenas os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 tiveram sucesso em atingir este alvo. Sendo que a probabilidade do algoritmo GR1F atingir o alvo é de 60% , passando para 75% no algoritmo GR1FPR-V0 e 100% no algoritmo GR1FPR-V1.

Destacamos então, que os algoritmos GRASP que utilizam o procedimento de Reconexão por Caminhos (GR1FPR-V0 e GR1FPR-V1) têm uma maior probabilidade de atingir os alvos médio e difícil.

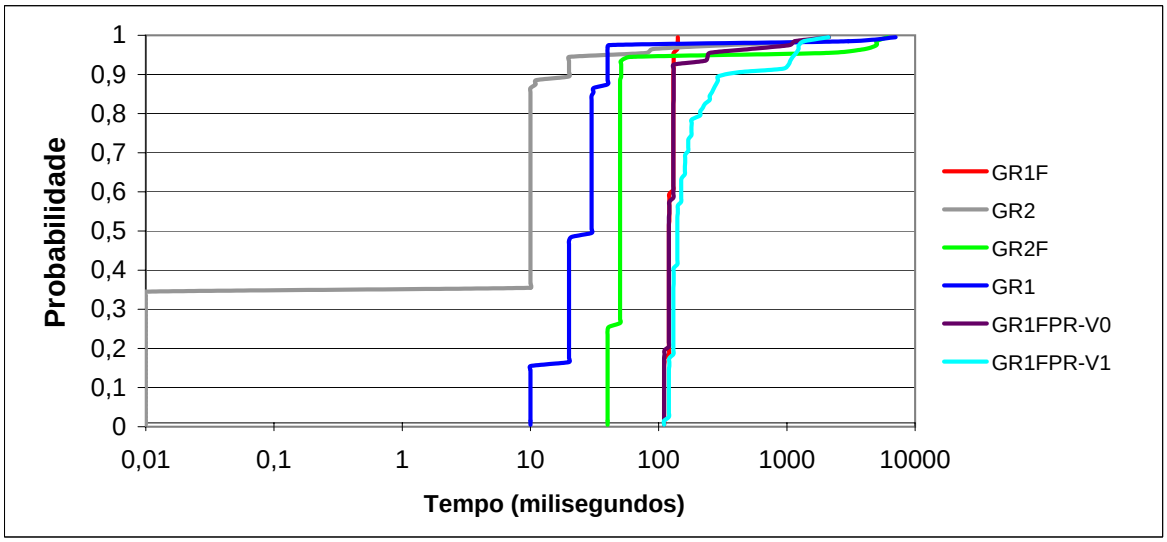

Gráfico 6.6 - Comparação entre os algoritmos GRASP - Problema 1001 (Grafo Original) - Alvo Fácil = 3858

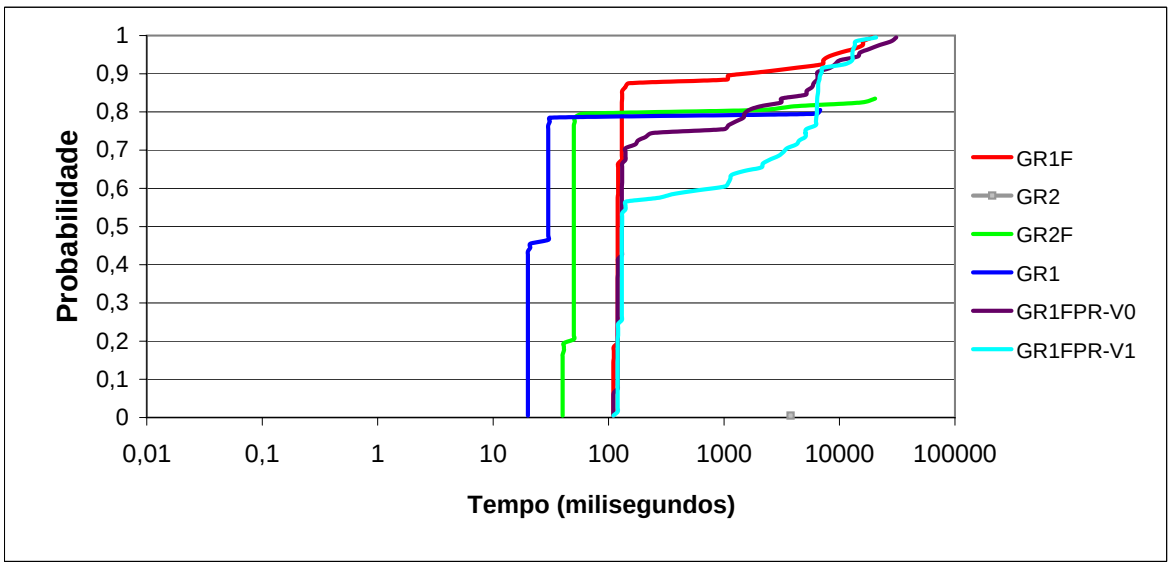

Gráfico 6.7 - Comparação entre os algoritmos GRASP - Problema 1001 (Grafo Original) - Alvo Médio = 3590

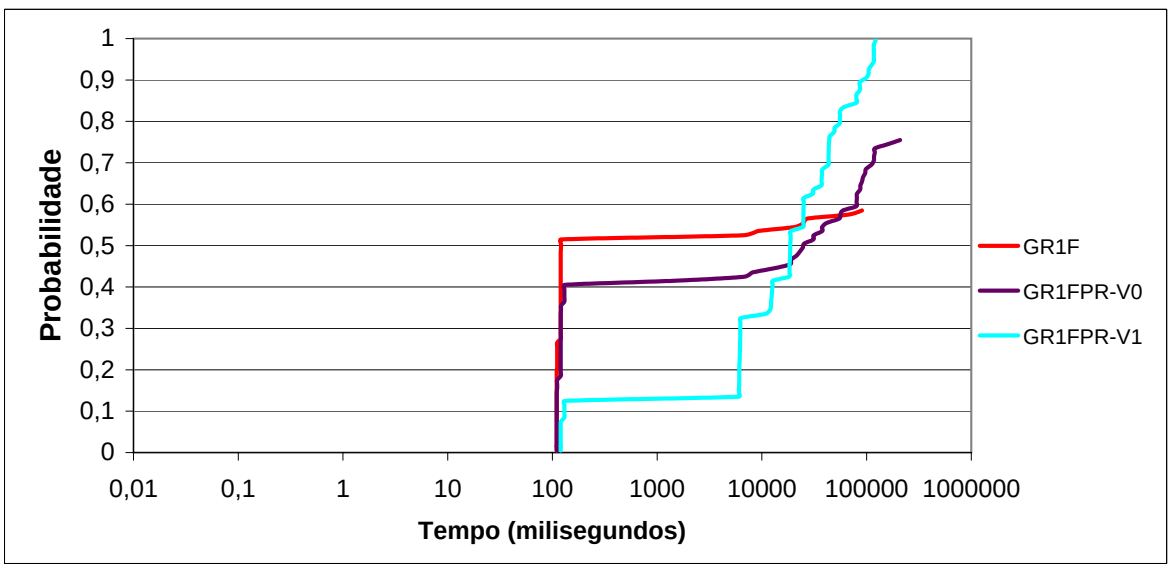

Gráfico 6.8 - Comparação entre os algoritmos GRASP - Problema 1001 (Grafo Original) - Alvo Difícil = 3396

Problema 1001 - Grafo Reduzido pelo Conjunto 1

No tempo de 100 milisegundos, os algoritmos GR1 e GR2 têm respectivamente a probabilidade de 100% e 97% de atingir o alvo fácil.

Quando consideramos o tempo de 20 milisegundos, observamos que os algoritmos GR1F,GR1FPR-V0,GR1FPR-V1 têm a probabilidade de 100% de atingir o alvo fácil.

O algoritmo GR2F encontrou o alvo fácil em quase todas as suas execuções (98%), consumindo um tempo de até 36000 milisegundos.

Ao considerarmos o alvo médio, o algoritmo GR1 apresentou uma melhor performance, quando comparado com os demais algoritmos, pois tem a probabilidade de quase 100% de atingir o alvo no tempo de 10 milisegundos. Enquanto, neste mesmo tempo, os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 têm a probabilidade em torno 45% de atingir o alvo.

Todavia, se considerarmos o tempo de 20 milisegundos, estes algoritmos têm a probabilidade de 100% de atingirem o alvo.

Confirmando as análises de desempenho anteriores, observamos que os algoritmos GR2 e GR2F são menos eficientes do que os demais algoritmos. Tendo em vista que estes algoritmos não atingiram o alvo médio em nenhuma das 100 execuções, deixamos ambos fora da análise.

Os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 têm de 95 % de probabilidade de atingir o alvo difícil no tempo de 20 milisegundos. Porém, apenas o algoritmo GR1FPR-V1 atinge este alvo em todas as execuções.

Podemos concluir que através do grafo reduzido, é possível alcançar os alvos médio e difícil consumindo um tempo inferior ao tempo consumido no grafo original. O que torna a convergência destes algoritmos mais rápida.

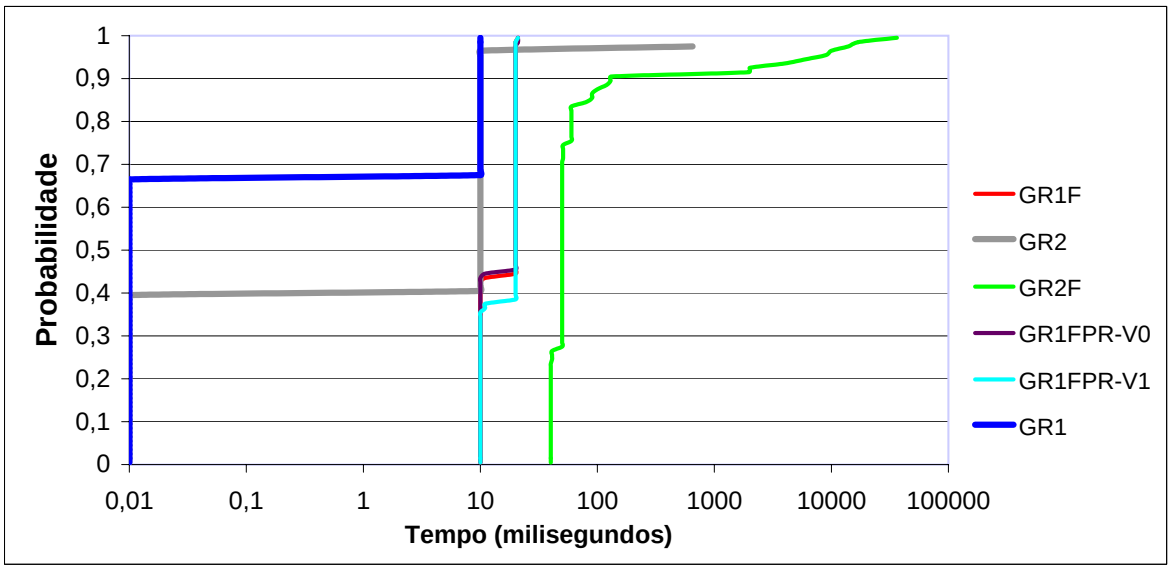

Gráfico 6.9 - Comparação entre os algoritmos GRASP - Problema 1001 (Red.Conj.1) - Alvo Fácil = 3365

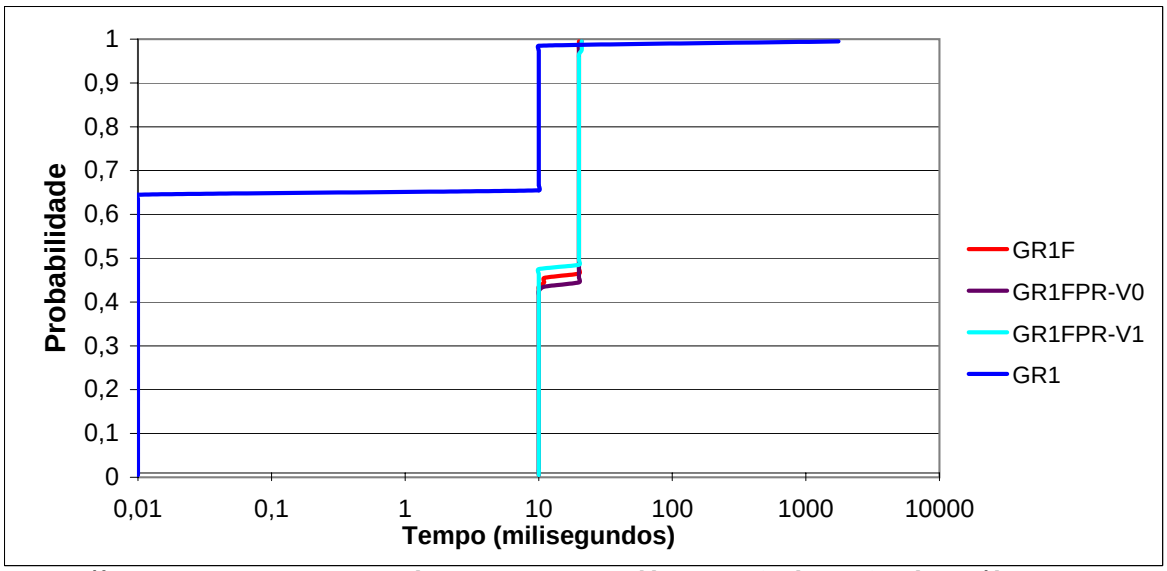

Gráfico 6.10 - Comparação entre os algoritmos GRASP - Problema 1001 (Red.Conj.1) - Alvo Médio = 2851

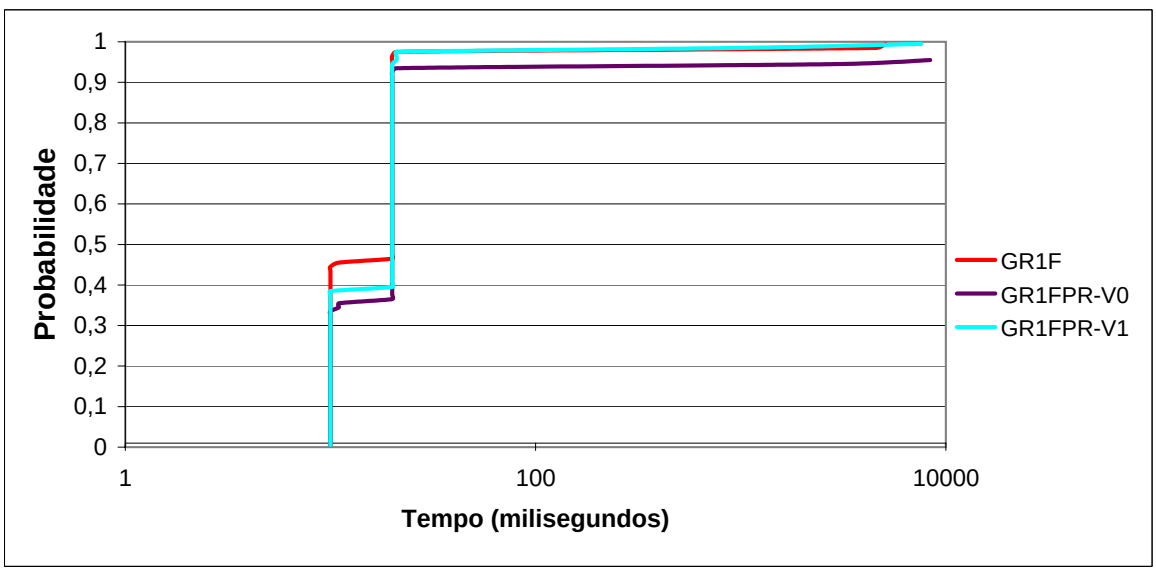

Gráfico 6.11 - Comparação entre os algoritmos GRASP - Problema 1001 (Red.Conj.1) - Alvo Difícil = 2611

Problema 1001 - Grafo Reduzido pelo Conjunto 2

Para os alvos fácil e médio, o algoritmo GR1 foi o que apresentou o melhor comportamento, pois atingiu estes alvos em 100% das execuções consumindo um tempo de no máximo 10 milisegundos. Já o algoritmo GR2, atinge estes alvos em 98% das vezes.

Os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 têm basicamente o mesmo comportamento, levando até 20 milisegundos para atingir o alvo fácil em 100% das execuções.

A probabilidade de que o algoritmo GR1F encontre o alvo médio em 10 milisegundos é de aproximadamente 45% , subindo para 50% no algoritmo GR1FPR-V0 e quase 55% no caso do algoritmo GR1FPR-V1.

Ao considerarmos o tempo de 20 milisegundos, todos os algoritmos acima citados, têm 100% de probabilidade de atingir o alvo médio. E no caso do algoritmo GR2F, a probabilidade de encontrar o alvo médio fica próxima de 80% no tempo de 10000 milisegundos.

Pelas considerações acima, podemos concluir que não há a necessidade de se utilizar algoritmos que têm os procedimentos de Filtro e Reconexão por Caminhos, quando consideramos o alvo fácil ou o alvo médio. Tendo vista que o algoritmo GR1 atinge estes alvos em tempo inferior aos demais algoritmos.

Quando consideramos o alvo difícil, o algoritmo GR1FPR-V1 foi o que apresentou o melhor comportamento, com a probabilidade de quase 100% de atingir o alvo, no tempo de até 20 milisegundos.

Independentemente do tipo de grafo (original, reduzido pelo conjunto 1 e pelo conjunto 2), os algoritmos GR2 e GR2F apresentaram a pior performance, quando comparados com os demais algoritmos. Acreditamos, que este fenômeno está vinculado ao tipo de heurística implementada na fase de busca local.

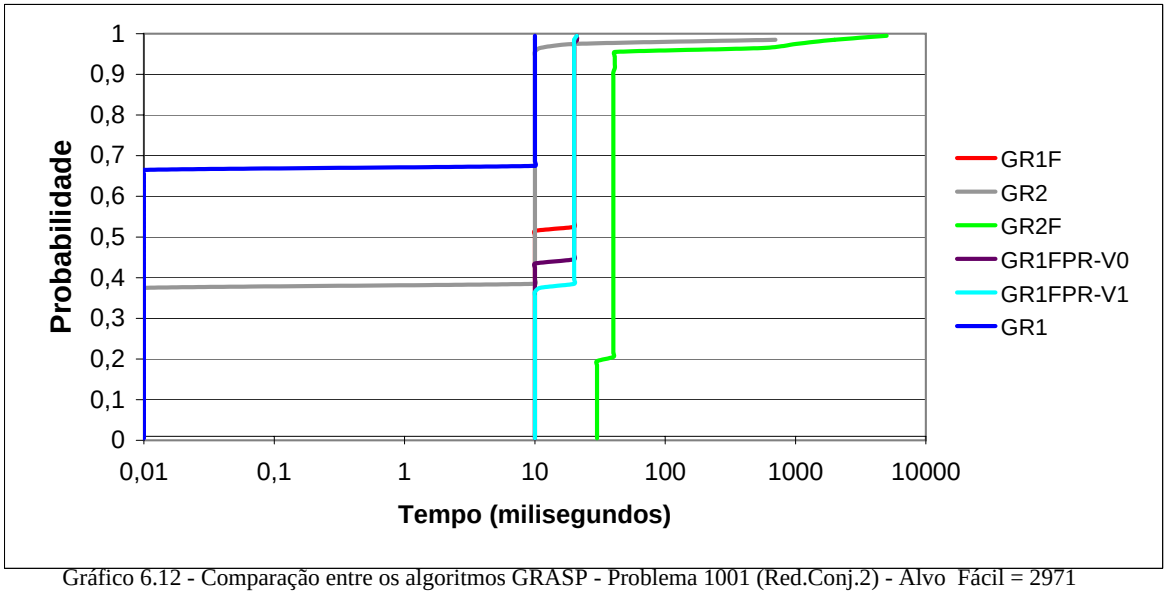

Gráfico 6.12 - Comparação entre os algoritmos GRASP - Problema 1001 (Red.Conj.2) - Alvo Fácil = 2971

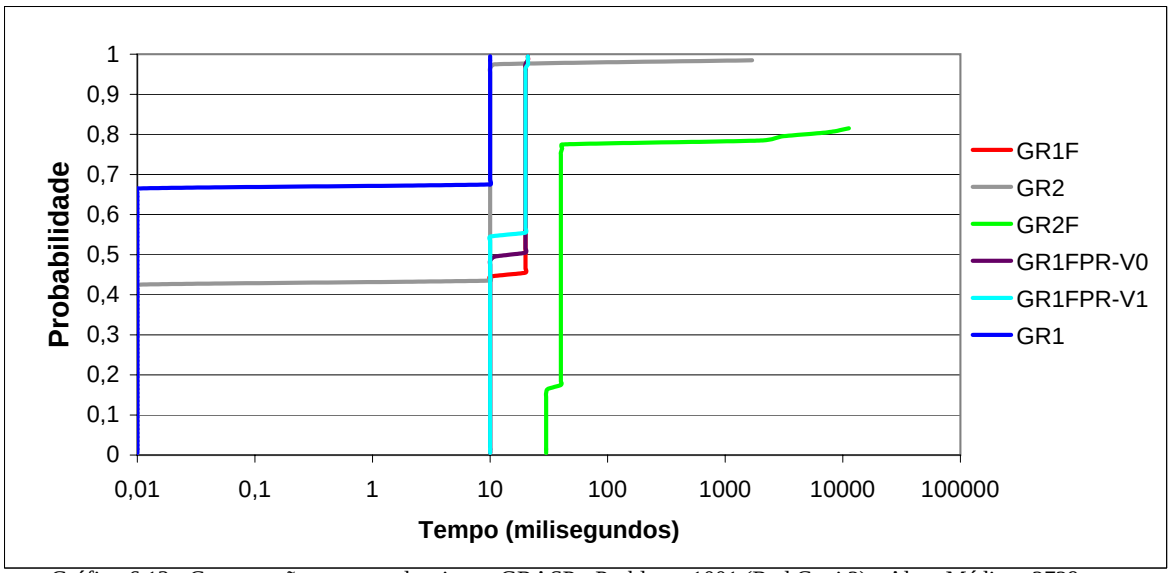

Gráfico 6.13 - Comparação entre os algoritmos GRASP - Problema 1001 (Red.Conj.2) - Alvo Médio = 2729

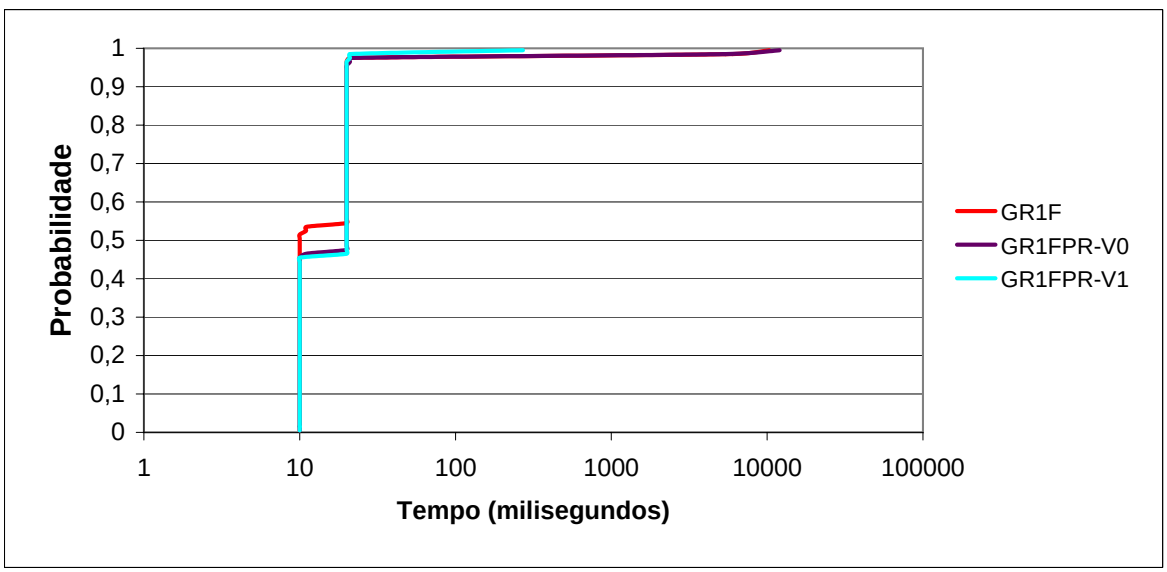

Gráfico 6.14 - Comparação entre os algoritmos GRASP - Problema 1001 (Red.Conj.2) - Alvo Difícil = 2612

Problema 2002 - Grafo Original

Tanto para o alvo fácil quanto para o alvo médio, o algoritmo GR1 apresentou uma melhor performance, quando comparado com os demais algoritmos. Em 100% das execuções este algoritmo alcançou estes alvos consumindo um tempo de no máximo 10 milisegundos.

Os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 apresentaram, praticamente, a mesma probabilidade (100%) de alcançar o alvo fácil em um tempo próximo de 70 milisegundos. E o algoritmo GR2F tem a probabilidade de 85% de encontrar o alvo fácil no tempo de 100 milisegundos e 100% de probabilidade de encontrar o alvo em um tempo superior a 10000 milisegundos.

Os algoritmos GR2 e GR2F não atingiram o alvo médio em nenhuma das 100 execuções. Apenas os algoritmos que possuem o VNS na sua fase de busca local (GR1, GR1F, GR1FPR-V0 e GR1FPR-V1) atingiram este alvo, em quase 100% das vezes, no tempo de até 80 milisegundos.

As probabilidades dos algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 atingirem o alvo difícil no tempo de aproximadamente 100 milisegundos são dadas respectivamente por: 30% , 55% e 70% .

O algoritmo GR1FPR-V1 alcança em quase 98% dos casos o alvo difícil no tempo de 63000 milisegundos. Enquanto o algoritmo GR1F atinge em um pouco mais de 30% das vezes este alvo.

Podemos concluir, que para um alvo mais difícil, há necessidade de utilizar um algoritmo que tenha Filtro e o procedimento de Reconexão por Caminhos, como é o caso dos algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1.

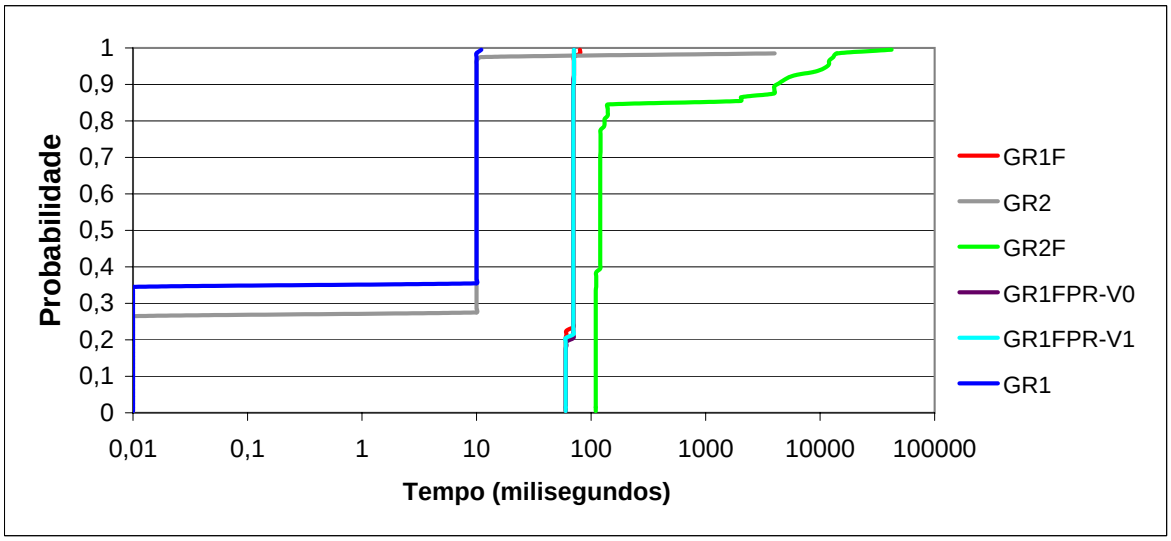

Gráfico 6.15 - Comparação entre os algoritmos GRASP - Problema 2002 (Grafo Original) - Alvo Fácil = 4826

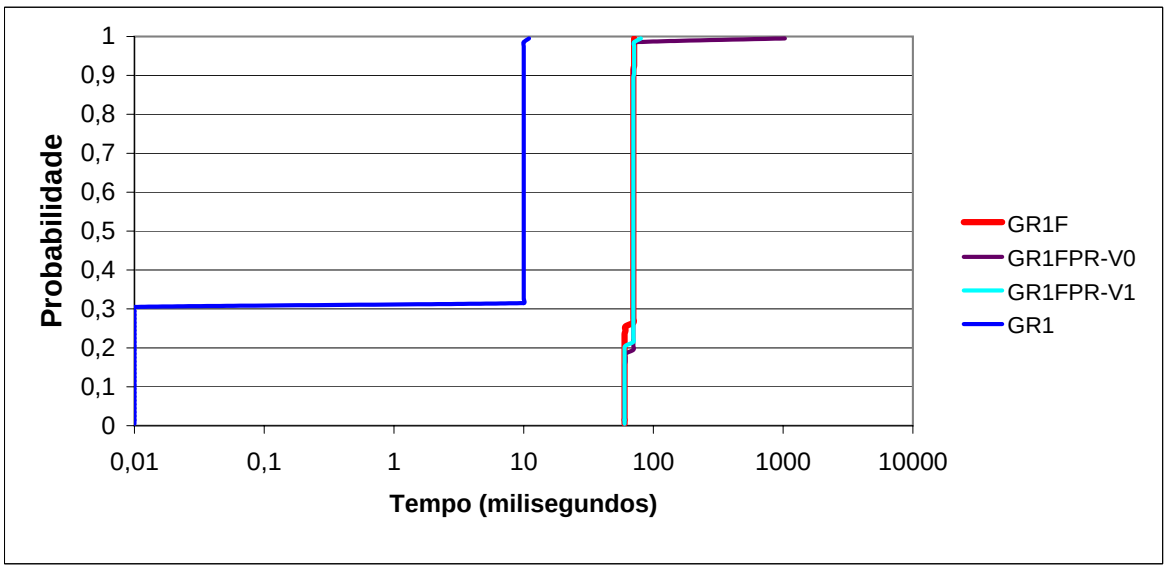

Gráfico 6.16 - Comparação entre os algoritmos GRASP - Problema 2002 (Grafo Original) - Alvo Médio = 3570

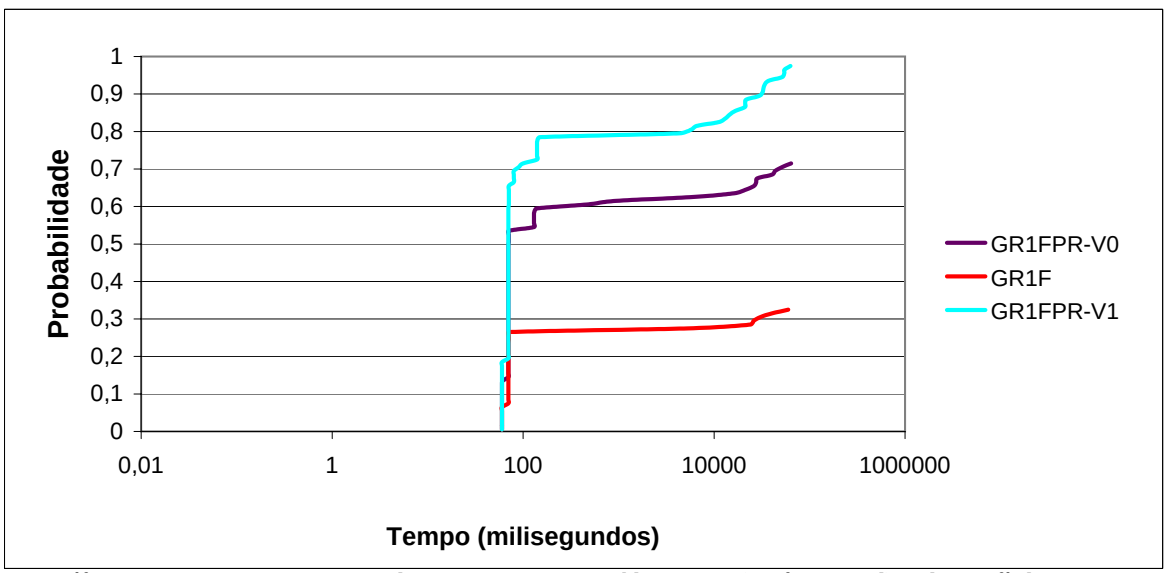

Gráfico 6.17 - Comparação entre os algoritmos GRASP - Problema 2002 (Grafo Original) - Alvo Difícil = 2939

Problema 2002 - Grafo Reduzido pelo Conjunto 1

Neste problema, considerando o alvo fácil, os algoritmos GRASP tiveram um comportamento similar ao observado para o problema 1001.

O algoritmo GR1 tem a probabilidade de 100% de atingir o alvo fácil no tempo próximo de 10 milisegundos. No caso dos algoritmos GR2 e GR2F, o tempo de processamento para alcançar este alvo em 100% das execuções foi bem maior, utilizando quase 2800 e 22000 milisegundos respectivamnente.

Os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 apresentam o mesmo comportamento, tendo 100% de probabilidade de atingir o alvo fácil no tempo de 20 milisegundos.

Novamente, no caso do alvo médio, o algoritmo GR1 apresenta um comportamento melhor que os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1.

Considerando o tempo de 10 milisegundos, a probabilidade do algoritmo GR1 encontrar o alvo médio é de 100% , caindo para 20% no algoritmo GR1FPR-V1 e 10% para os algoritmos GR1FPR-V0 e GR1F.

Da mesma forma que no problema 1001 (conjunto 1), observamos mais uma vez, que os algoritmos GR2 e GR2F não atingiram em nenhuma das 100 iterações o alvo médio .

É notável que apenas os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 alcancem o alvo difícil em todas as execuções. Estes algoritmos têm probabilidade acima de 90% de encontrarem o alvo no tempo de 20 milisegundos.

Nenhum destes algoritmos atingiram este alvo nas 100 vezes, porém o algoritmo GR1FPR-V0 apresenta o melhor comportamento quando comparado com os outros dois algoritmos.

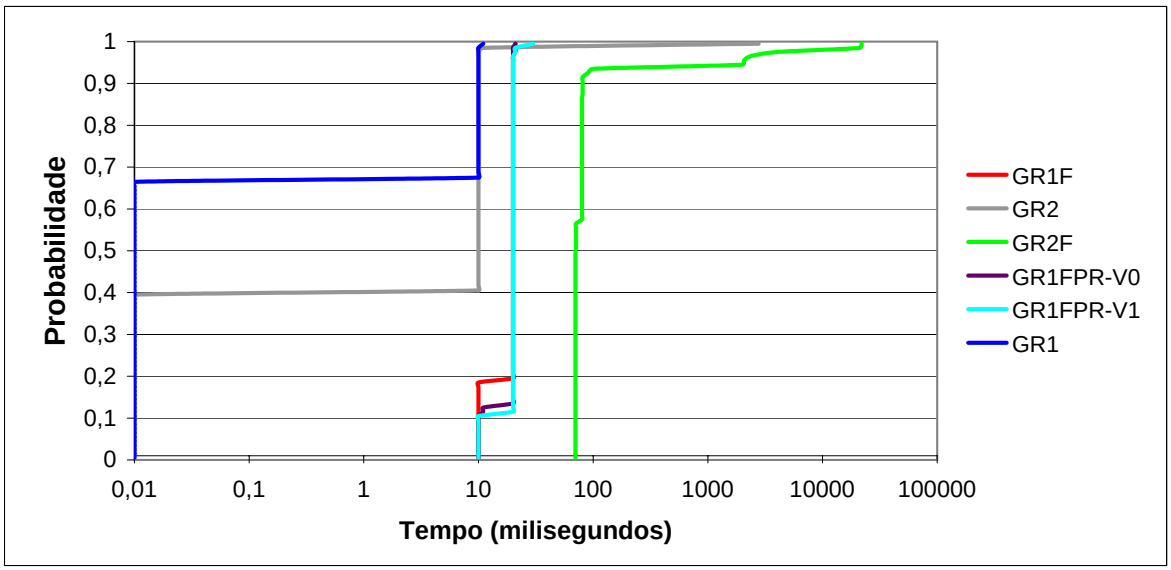

Gráfico 6.18 - Comparação entre os algoritmos GRASP - Problema 2002 (Red.Conj.1) - Alvo Fácil = 5469

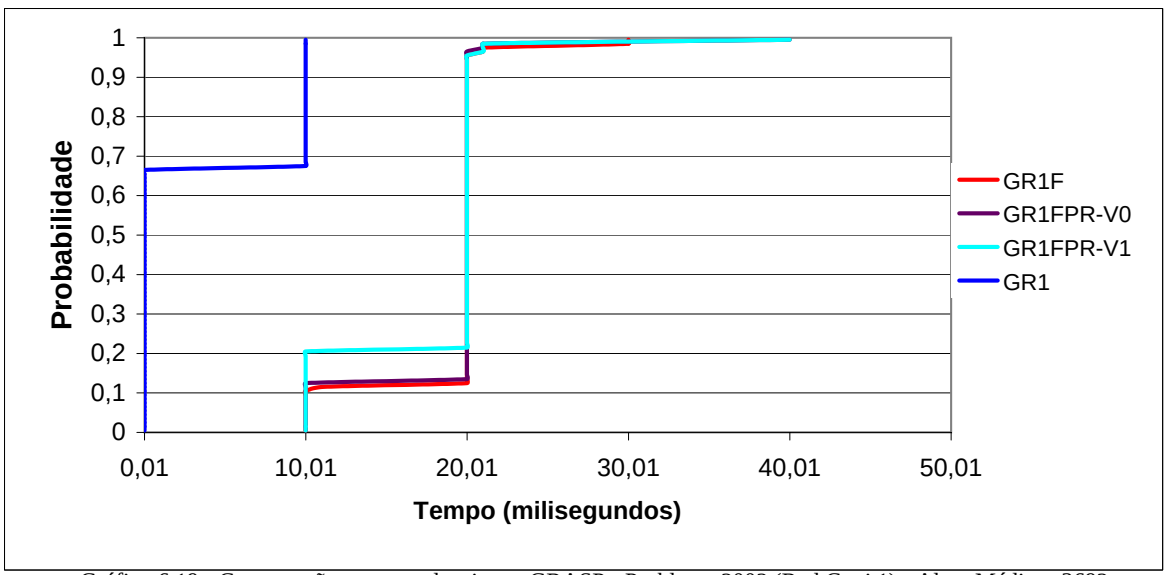

Gráfico 6.19 - Comparação entre os algoritmos GRASP - Problema 2002 (Red.Conj.1) - Alvo Médio = 3692

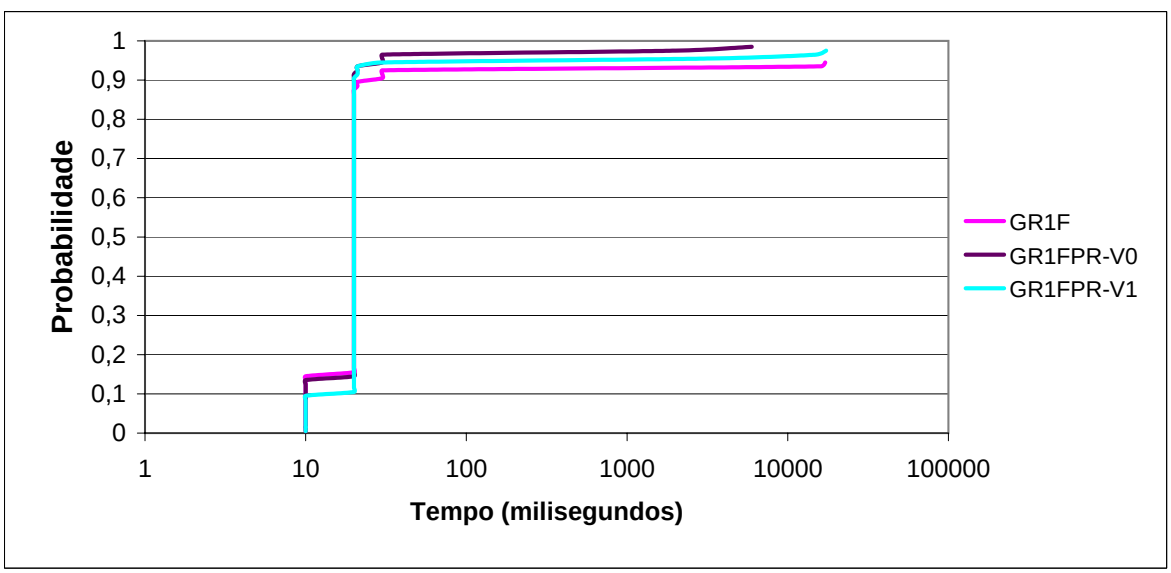

Gráfico 6.20 - Comparação entre os algoritmos GRASP - Problema 2002 (Red.Conj.1) - Alvo Difícil = 2830

Problema 2002 - Grafo Reduzido pelo Conjunto 2

Considerando o tempo de 0,01 milisegundos, apenas os algoritmos GR2 e GR1 atingem o alvo fácil em 40% e 65% de suas execuções.

A probabilidade que o algoritmo GR1 tem de encontrar o alvo fácil no tempo de 10 milisegundos é de quase 100% , caindo para 95% no algoritmo GR2 e aproximadamente 55% nos algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1.

Quando consideramos o alvo médio, apenas os algoritmos GR1F, GR1FPR-V0 e GR1FPR-V1 têm 100 % de probabilidade de atingir o alvo no tempo de 20 milisegundos. Já o algoritmo GR2F, atinge o alvo médio em apenas 80% de suas execuções.

Os algoritmos GR1, GR1FPR-V0 e GR1FPR-V1 apresentam praticamente o mesmo comportamento. Estes 3 algoritmos têm quase 100% de probabilidade de atingir o alvo difícil no tempo de 20 milisegundos.

Conclusões gerais sobre a análise probabilística dos problemas 1001 e 2002:

Após analisarmos cada um dos algoritmos, considerando todos os alvos, podemos observar que os algoritmos GR1 e suas variações (GR1F, GR1FPR-V0 e GR1FPR-V1) apresentaram uma performance melhor que os algoritmos GR2 e GR2F.

Isto reforça a observação feita a partir da análise das soluções dos 70 problemas teste e dos gráficos das soluções. Ou seja, há um boa indicação, que estes algoritmos terão comportamentos similares para outros problemas.

Acredita-se que o bom desempenho dos algoritmos acima citados seja decorrente da utilização da metaheurística VNS na fase de busca local e do procedimento de Reconexão por Caminhos.

Para os alvos difíceis, o algoritmo GR1FPR-V1 apresentou a melhor performance, decorrente provavelmente, da utilização da reconexão por Caminhos após a fase de busca local.

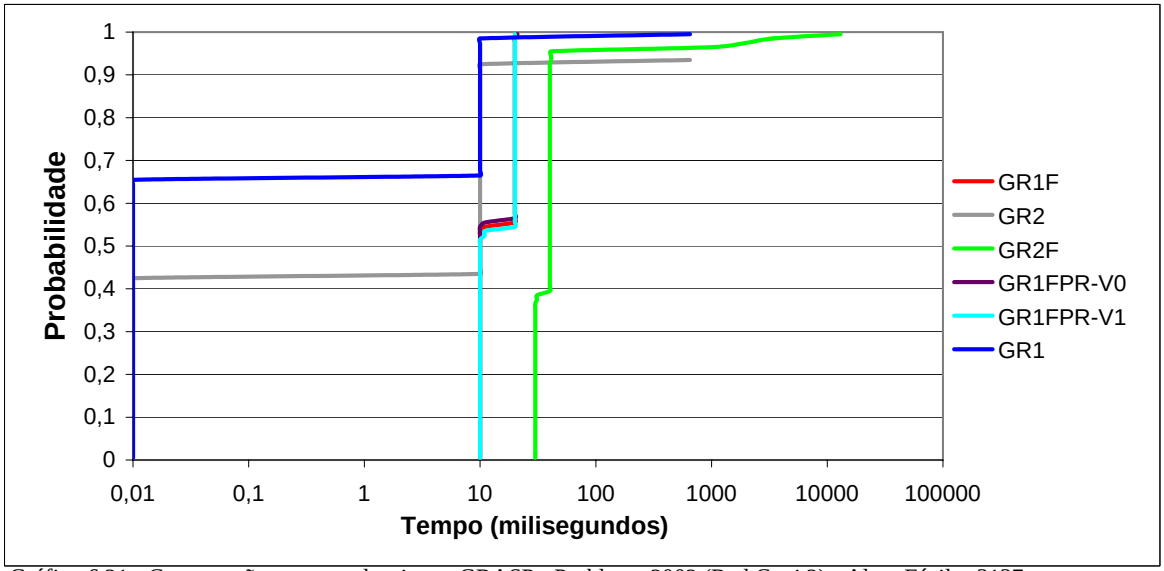

Gráfico 6.21 - Comparação entre os algoritmos GRASP - Problema 2002 (Red.Conj.2) - Alvo Fácil = 3137

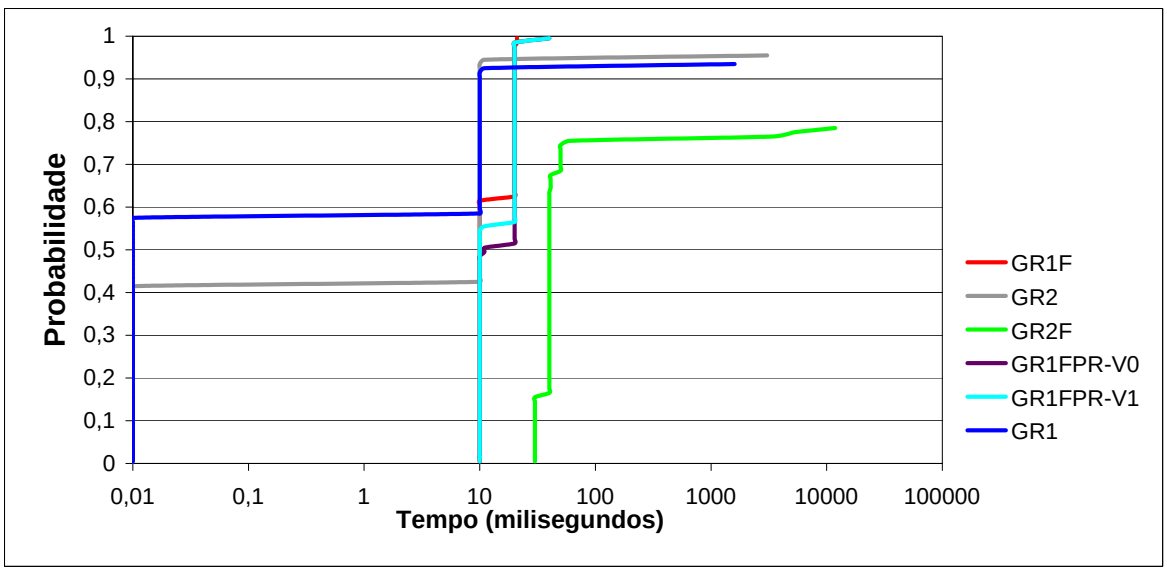

Gráfico 6.22 - Comparação entre os algoritmos GRASP - Problema 2002 (Red.Conj.2) - Alvo Médio = 2938

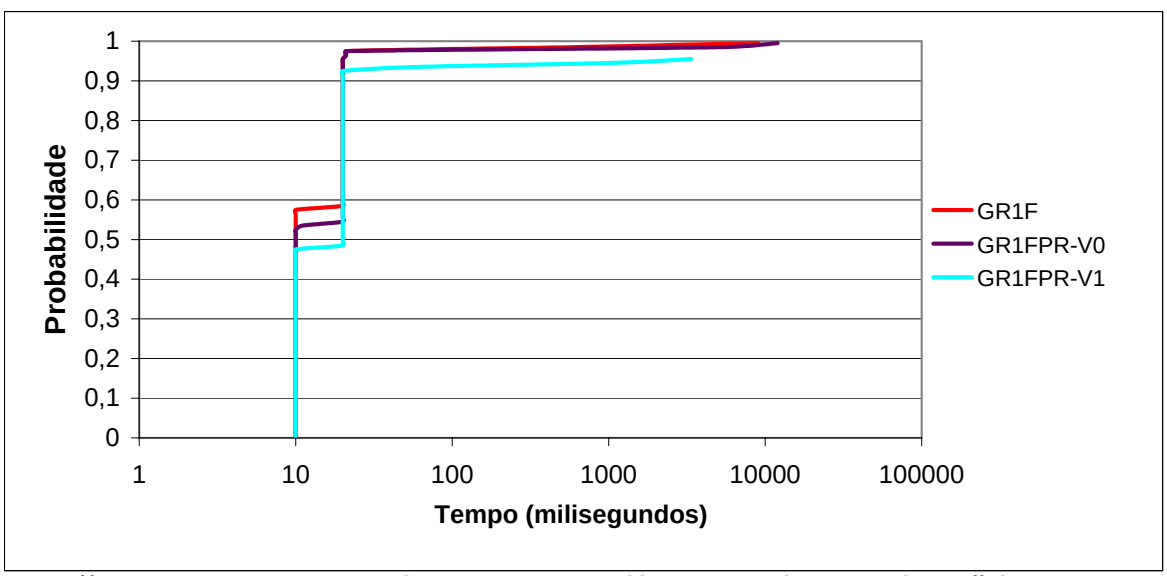

Gráfico 6.23 - Comparação entre os algoritmos GRASP - Problema 2002 (Red.Conj.2) - Alvo Difícil = 2814

7 - Conclusões

Conforme observado ao longo deste trabalho, o Problema de Recobrimento Rotas é um problema de otimização combinatória de difícil solução e que está associado a um vasto conjunto de aplicações reais.

Tendo em vista a dificuldade de resolução do PRR e com o intuito de auxiliar na solução de um maior número de problemas, desenvolvemos as seguintes metodologias alternativas:

- Uma nova formulação matemática que se diferencia das demais formulações existentes na literatura quanto às restrições que asseguram a não formação de ciclos desconexos da origem. Estas restrições estão diretamente ligadas às variáveis de fluxo que associamos a cada aresta do grafo, relacionado a uma instância do Problema de Recobrimento de Rotas.

Com a utilização desta nova formulação, conseguimos obter solução ótima para todos problemas de até 60 vértices e alguns problemas de 70, 80, 100 e 200 vértices.

- Há uma boa indicação de que a nova formulação proposta para o PRR, pode ser mais eficiente do que a formulação de Maniezzo. Esta consideração esta baseada na comparação dos tempos de processamento destas duas formulações, analisando um conjunto de 20 problemas teste.

- Uma nova regra de redução, chamada de Regra Nova, que em conjunto com as regras 3 e 4 da literatura nos fornece grafos de menor tamanho em relação ao grafo original. A partir destas três regras de redução, definimos 2 conjuntos de regras, assim denominados: Conjunto 1 é formado pelas regras 3 e 4 da literatura e o conjunto 2 formado pela Regra Nova e pelas regras 3 e 4.

Com a aplicação destes dois conjuntos de regras nos grafos originais, produzimos grafos reduzidos que foram utilizados como dados de entrada para a nova formulação. E desta forma, foi possível obter soluções ótimas para problemas teste de até 80 vértices e alguns problemas com 100 vértices e 200 vértices.

Observamos ainda, que utilizando estes grafos reduzidos nos algoritmos GRASP, foi possível obter soluções viáveis de qualidade superior (menores limites superiores) para o PRR, quando comparadas com as soluções obtidas a partir do grafo original.

Tendo em vista, que as regras 1 e 2 da literatura podem produzir rotas inviáveis para o grafo original associado ao PRR, não incorporamos estas regras aos conjuntos 1 e 2.

• Desenvolvimento de uma relaxação lagrangeana a partir da nova formulação proposta para o PRR. Nesta relaxação, agregamos à função objetivo as restrições do tipo $\sum_{v_k \in S_l} y_k \geq 1 \quad (\forall\, w_l \in W)$ que asseguram que cada vértice do conjunto W seja coberto pelo menos por um dos vértices do conjunto V.

Ao utilizarmos esta relaxação, tivemos a possibilidade de gerar limites inferiores melhores do que aqueles obtidos mediante a aplicação da relaxação linear (Bazaraa[27]).

• A implementação de uma heurística lagrangeana simples, considerando as particularidades do PRR. Com a utilização da heurística lagrangeana, "ajustamos " a solução obtida a partir da relaxação lagrangeana, de forma a obter uma solução viável, ou seja, um limite superior para a solução ótima do PRR.

• Desenvolvimento de um conjunto de seis algoritmos baseados nas metaheurísticas GRASP e VNS.

Estes algoritmos, possibilitaram a obtenção de soluções viáveis para diversos problemas teste de até 500 vértices, associados ao PRR. Em vários destes problemas, não havia sido possível conseguir nem mesmo uma solução viável, quando utilizamos a nova formulação.

• Efetuamos uma análise das soluções dos seis algoritmos GRASP considerando um conjunto de 70 problemas teste com número de vértices variando de 10 a 500.

Através desta análise, tivemos a oportunidade de verificar, que os algoritmos GR1FPR-V0 e GR1FPR-V1 produziram soluções viáveis de melhor qualidade, quando comparados com os demais algoritmos.

• Desenvolvimento de uma análise probabilística utilizando os problemas 1001 (100 vértices) e 2002 (200 vértices). Foram estabelecidos para cada um destes problemas três tipos de alvo: fácil, médio e difícil. Considerando os grafos original, reduzido pelo conjunto 1 e reduzido pelo conjunto 2.

A partir da análise probabilística foi possível observar os seguintes fatos:

(1) Os algoritmos GR1FPR-V0 e GR1FPR-V1 têm uma maior probabilidade de alcançar os alvos difíceis (média das soluções do melhor algoritmo). Estes algoritmos são mais sofisticados que os demais, pois além do procedimento de Filtro, utilizam o VNS e a Reconexão por Caminhos.

(2) Independentemente do tipo de grafo (original, reduzido pelo conjunto 1 e pelo conjunto 2), os algoritmos GR2 e GR2F apresentaram a pior performance, quando comparados com os demais algoritmos.

Acreditamos, que este fenômeno está vinculado à implementação de uma heurística simples na fase de busca local.

Em síntese, pelas observações feitas, podemos concluir que a partir da utilização das novas metodologias propostas neste trabalho de tese, pode-se dar um boa contribuição para resolução do Problema de Recobrimento de Rotas.

A seguir, apresentamos os trabalhos que futuramente podem ser propostos como desdobramentos deste trabalho de tese:

- Criação de algoritmos mais eficazes utilizando novas formas de religamento (Path Relinking) e considerando o tamanho da Lista de Candidatos Restrita variado (GRASP Reativo);

- Adaptação das metodologias propostas neste trabalho para resolver algumas variações do PRR: Problema de Recobrimento com Múltiplas Rotas (m-PRR), que consiste em determinar m rotas de comprimento mínimo, todas partindo e chegando a um mesmo vértice origem (Hachicha *et al.*[18]).

O Problema de Recobrimento de Rotas com Time-Windows (PRRTW) (Desrochers *et al.*[20], Solomon[21] e Solomon *et al.*[22]), onde os vértices do conjunto V possuem um intervalo de tempo para serem visitados. Seja (e_i, l_i) , a janela de tempo (Time-Windows) do vértice $v_i \in V$, onde e_i é o tempo mínimo para visitar o vértice v_i e l_i é o tempo máximo para visitar o vértice v_i .

- Estudar novas relaxações e heurísticas lagrangeanas para resolução do PRR.

8 - Bibliografia

[1] GENDREAU, M., LAPORTE, G. and SEMET, F., "The Covering Tour Problem ", *Operations Research*, v. 45, pp. 568-576, 1995.

[2] SWARFICTER, J. L., *Grafos e Algoritmos Computacionais*, 2^a edição, Editora Campus, 1998.

[3] WOLSEY, L. A., "Notes on Integer Programming ", *CORE et Faculté des Sciences Appliquées, Université Catholique de Louvain*, 1997.

[4] LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G., "The Traveling Salesman Problem ", *Wiley-Interscience Series in Discret Mathematics*, 1997.

[5] CURRENT, J. R., *Multiobjective Design of Transportation Networks*, Ph.D. Thesis, Department of Geography and Environmental Engineering, The Johns Hopkins University, 1981.

[6] GENDREAU, M., HERTZ, A. and LAPORTE, G., "New Insertion and Postoptimization Procedures for The Traveling Salesman Problem ", *Opns. Res*, v. 40, pp. 1086-1094, 1992.

[7] BALAS, E. and HO, A., "Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optmization: A Computational Study ", *Math. Programming*, v.12, pp. 37-70, 1980.

[8] MANIEZZO, V., BALDACCI, R. and ZAMBONI, M., "Scatter Search Methods for The Covering Tour Problem ", *Scienze dell Informazione*, University of Bologna, 1999.

[9] GLOVER, F., "Scatter Search and Star Paths: Beyond the Genetic Metaphor", *OR. Spektrum*, v. 17, pp. 125-137, 1995.

[10] REEVES, C., "Modern Heuristic Techniques for Combinatorial Problems ", *Blackwell Scientific Publications*, 1993.

[11] FINKE, G., CLAUS, A. and GUNN, E., "A Two-Commodity Network Flow Approach to the Traveling Salesman Problem ", *Congress. Numerantium*, v. 41, pp. 167-178, 1984.

[12] CURRENT, J.R. and SCHILLING, D. A., "The Covering Salesman Problem ", *Transportation Science*, v. 23, pp. 208-213,1989.

[13] CURRENT, J.R. and ROLLAND, E., "Efficient Algorithms for Solving the Shortest Covering Path Problem ", *Transportation Science*, v.28, pp. 317-327, 1994.

[14] BALAS, E., "The Prize Collecting Traveling Salesman Problem ", *ORSA/TIMS*, 1986.

[15] FISCHETTI, M. and TOTH, P., "An Additive Approach for the Optimal Solution of the Prize Collecting Traveling Salesman Problem, in Vehicle Routing: Methods and Studies ". In: B.L. Golden and A.A. Assad (eds.), pp. 319-343, North-Holland, Amsterdam, 1988.

[16] LAPORTE, G. and MARTELLO, S., "The Selective Traveling Salesman Problem ", *Discrete Appl. Math.*, v. 26, pp. 193-207, 1990.

[17] CURRENT, J. R and SCHILING, D. A, "The Median Tour Problem and Maximal Covering Tour Problems: Formulations and Heuristics ", *European Journal of Operational Research*, v.73, pp. 114-126, 1994.

[18] HACHICHA, M., HODGSON, M. J., LAPORTE, G., "*et al.* ", "Heuristics for the Multi-Vehicle Covering Tour Problem ", *Computers and Operations Research*, v. 27, pp. 29-42, 2000.

[19] MOTTA, L., OCHI, L. S. and MARTINHON, C., "GRASP Metaheuristics to the Generalized Covering Tour Problem ". *In: Proceedings of the IV Metaheuristic International Conference (IV-MIC2001)*, Porto, Portugal.

[20] DESROCHERS, M., LENSTRA, J. K., SAVELSBERGH, M. W. P., "*et al.*", " Vehicle Routing with Time Windows: Optimization and Approximation, in Vehicle Routing: Methods and Studies ". In: B.L. Golden and A.A. Assad (eds.), pp. 65-84, North-Holland, Amsterdam, 1988.

[21] SOLOMON, M. M., "Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints ", *Operations Research*, v. 35, pp. 254-265, 1987.

[22] SOLOMON, M. M., BAKER, E. K. and SCHAFFER, J. R., "Vehicle Routing and Scheduling Problems with Time Windows Constraints: Efficient Implementations of Solution Improvement Procedures, in Vehicle Routing: Methods and Studies ". In: B.L. Golden and A.A. Assad (eds.), pp. 85-106, North-Holland, Amsterdam, 1988.

[23] CHISMAN, J. A., "The Clustered Traveling Salesman Problem ", *Comput. and Ops. Res*, v. 2, pp. 115-119, 1975.

[24] ANILY, S., BRAMEL, J., and HERTZ, A., "A $\frac{5}{3}$-approximation Algorithm for the Clustered Traveling Salesman Tour and Path Problems ", *Operations Research Letters*, v. 24, pp. 29-35, 1999.

[25] MACULAN, N. e CAMPELO, R. E., *Algoritmos e Heurísticas*, EDUFF-Niterói, 1994.

[26] GEOFFRION, A.M., "Lagrangean Relaxation and its use in Interger Programming", *Mathematical Programming Study*, v. 2, pp. 82-114, 1974.

[27] BAZARAA, M. S., *Linear Programming Networks Flows*, Jonh Wiley e Sons, 1993.

[28] HELD, M. and KARP, R.M., "The Travelling Salesman Problem and Minimum Spanning Trees ", *Ops. Res.,* v.18, pp. 1138-1162.

[29] BRITO, L. R., MAKLER, S. S., e OCHI, L. S., "Uma Nova Regra de Redução para o Problema de Recobrimento de Rotas ". In: Anais do XXXIII Simpósio Brasileiro de Pesquisa Operacional (XXXIII SBPO), pp. 1307-1316, ISSN 1518-1731, em CD-ROM, Campos do Jordão, SP, 2001.

[30] VIANA, V., "Meta-Heurística e Programação Pararela em Otimização Combinatória ", UFC edições, 1998.

[31]GLOVER, F., "Tabu Search ". *Norwell*, MA: Kluwer Academic Publishers, 1997.

[32] HANSEN, P. and MLADENOVIĆ, N., "Variable Neighborhood Search for the p-Median ", *Location Science*, v. 5, pp. 207-226, 1997.

[33] HANSEN, P., MLADENOVIĆ, N. and PEREZ-BRITO, D., "Variable Neighborhood Decompositon Search ", GERAD, 1998.

[34] RESENDE, M. G. C. and FEO, T. A., "Greedy Randomized Adaptative Search Procedures ", *Journal of Global Optimization*, pp. 1-27, 1995.

[35] LAGUNA, M. and MARTI, R., "GRASP and Path-Relinking for 2-layer Straight Line Crossing Minimization ", *INFORMS Journal on Computing*, v.11, pp. 44-52, 1999.

[36] AIEX, R. M., RESENDE, M. G. C., PARDALOS, P. M. and TORALDO, G., "GRASP with Path Relinking for the Three-Index Assignment Problem ", *Technical Report* NJ 07932: AT&T Labs Research, Florham Park, 2000.

[37] RESENDE, M. G. C. and RIBEIRO, C. C., "GRASP with Path- Relinking: Recent Advances and Applications ", *AT&T Labs Research Technical Report* , 2003.

[38]GOLDBARG, M.C, LUNA, H. P., , "Otimização Combinatória e Programação Linear: Modelos e Algoritmos ", Editora Campus, Rio de Janeiro, 2000.

[39] DASH, BISWOTH and NORTHAANTS *XPRESS-MP*, NN73BX, UK, 1994.