Novas Abordagens para o Problema de Recobrimento de Rotas

Luciene Cristina Soares Motta

Dissertação apresentada ao Curso de Pós-Graduação
em Ciência da Computação da Universidade Federal Fluminense
como parte dos requisitos para a obtenção do título de
Mestre em Ciência da Computação.

Orientador: Luiz Satoru Ochi
Co-orientador: Carlos Alberto de Jesus Martinhon

Fevereiro de 2001

Aos meus pais, irmãos e avós por conseguirmos
fazer de nossa família o porto onde sempre
queremos ancorar o nosso barco.
Luciene C. S. Motta

Agradecimentos

À Deus, por ter me fortificado na fé nos diversos momentos em que o desânimo se fez presente;

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro concedido através da minha bolsa de estudos;

Ao Instituto de Computação da Universisdade Federal Fluminense, por ter dado a oportunidade de desenvolver meus estudos;

Ao Professor Nelson Maculan da Universidade Federal do Rio de Janeiro, por ter concedido o laboratório de otimização da COPPE/Sistemas para realizarmos os testes com o XPRESS.

Ao meu orientador Luiz Satoru Ochi, pela sua amizade, dedicação e pelo crescimento profissional e pessoal que me proporcionou nestes anos em que nos dedicamos a este trabalho;

Ao meu co-orientador Carlos Alberto de Jesus Martinhon, pela credibilidade, compreensão e pelo incentivo presentes neste período de orientação;

Aos Professores Cláudia Linhares Sales, da Universidade Federal do Ceará; Marcone J. Freitas de Souza, da Universidade Federal de Ouro Preto e Valdomiro Neves Lima, da Universidade Federal Rural do Rio de Janeiro, por aceitarem o convite para fazerem parte da banca examinadora deste trabalho;

Ao meu eterno Professor Valdomiro Neves Lima, por ter acreditado na minha capacidade e por ter semeado este mestrado. O meu muito obrigado por ter estado sempre presente em todas as etapas deste trabalho, independente da distância e do contato pouco freqüente.

À Professora Cláudia Linhares Sales, pelo seu brilho inerente, amor e entusiasmo dedicados ao seu trabalho que me contagiaram durante o nosso convívio, sem os quais minha formação acadêmica ficaria incompleta.

Ao Professor Marcone J. Freitas de Souza, pelo empenho destinado para que os testes com o XPRESS acontecessem;

Às professoras Cristina Boeres e Lúcia Drummond pelas susgestões fornecidas na fase de propostas deste trabalho;

Aos Professores Carlos Moura a Ana pelo incentivo sempre muito carinhoso e pelo profissionalismo com que desempenham suas atividades;

À todos os funcionários do Instituto de Computação, em especial à Ângela e ao Carlinhos pela boa vontade em "quebrar todos os nossos galhos";

Ao meu grande amigo e eterno parceiro Valdir, por ter contribuído de forma única para o meu crescimento pessoal e para meu sucesso nos estudos, estando sempre com a mão estendida mesmo nas horas em que eu sequer percebesse. À Wal, João e Pedro Henrique pela maravilhosa acolhida;

À Dani, pelo carinho, incentivo, amizade e pelo doce refúgio em que a sua casa se transformou na reta final deste trabalho. Obrigada pelos cafés bem fortes e pelas nossas horas de conversas!

Ao meu amigo Mozar, pela ajuda na implementação, pela sua compania na véspera da defesa e pelos "cineminhas";

À minha grande amiga e mãezona Cândida, que mesmo distante sempre esteve presente no meu coração;

À um amigo mais do que especial, Rônnet, que durante oito anos esteve na "primeira fileira" me incentivando e amparando a cada passo novo. O meu carinho ao meu "paizinho" Beto, Jannet, Rodrigo e vó Odete;

Ao Gustavo, pelas nossas longas horas de "devaneios" que tanto contribuíram para o meu auto-conhecimento;

Aos amigos que o mestrado me proporcionou: Hans, David, Alberto, Robson, Carlos França, Silo, Jacques, Sônia, Leonardo Tadeu, Leonardo Vieira, Fábio, Baliu e Guilherme;

Aos amigos de todos os momentos Zê, Rica e Cleber, pela constante torcida;

Finalmente, o meu "muito obrigada" mais especial, aos meus pais Lêda e Ilmar pelo meu caráter e pelo amor incondicional, aos meus irmãos Letícia, Lucimar e Leonardo pela nossa amizade e cumplicidade sem igual, às minhas vozinhas Jurandy e Andreza pelas suas incansáveis orações e pelo carinho, ao meu cunhado Marcelo ao Gustavo Iglesias. Amo vocês!!!

"Aquele que se sente incomodado pelo ignorante
esquece-se que muitos talvez não os fossem se
tivessem as mesmas luzes que iluminam
o caminho de uns poucos"
Marco Natali

Abstract

The Covering Tour Problem is a job sequencing problem and it is defined on a graph $G = (V \cup W, E)$, where W is a set of vertices that must be covered. The problem consists of determining a minimum length Hamiltonian cycle on a subset of V such that every vertex of W is within a distance d from at least one node in the cycle. Being a generalization of the Traveling Salesman Problem, this problem is NP-Hard. This work presents a new mathematical formulation based on flow variables, reduction rules for the associated graph and original metaheuristic algorithms to solve a generalized version of the Covering Tour Problem approximately.

Resumo

O Problema de Recobrimento de Rotas (PRR) é um problema de seqüenciamento de tarefas definido sob um grafo $G = (V \cup W, E)$ onde W é o conjunto de vértices que devem ser cobertos. O problema consiste em determinar uma rota ou um ciclo Hamiltoniano de comprimento mínimo sob um subconjunto de V, de modo que todo vértice de W diste no máximo d de algum vértice da rota. Sendo uma generalização do Problema do Caixeiro Viajante (PCV), o PRR é considerado NP-Difícil. Este trabalho apresenta uma nova formulação matemática baseada em variáveis de fluxo, regras para a redução do grafo associado e metaheurísticas para solucionar aproximadamente uma versão generalizada do Problema de Recobrimento de Rotas (PRRG).

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O impacto das novas tecnologias em computação tem apresentado uma crescente necessidade de se resolver problemas cada vez mais difíceis. Muitos destes, são problemas de otimização que englobam a modelagem de aplicações reais, onde os métodos clássicos de otimização têm encontrado grandes dificuldades para obter uma solução ótima, mesmo para os que possuem, teoricamente, a garantia de atingí-la.

A grande quantidade de problemas de otimização de elevada complexidade encontrados em diferentes áreas, tem provocado a necessidade de se pesquisar o desenvolvimento de métodos mais eficientes, não só do ponto de vista teórico, como também do ponto de vista operacional. Esses métodos surgem usualmente do resultado da adaptação de idéias de uma grande variedade de áreas, esperando que se produza procedimentos eficientes e flexíveis para lidar com problemas cada vez mais complexos e dinâmicos. Algumas dessas idéias têm motivado o surgimento de estruturas com novas metodologias, como por exemplo a dos *Algoritmos Genéticos* [1], que procuram imitar o fenômeno biológico da reprodução evolutiva; as *Colônias de Formigas* [2], que simulam o comportamento das formigas que cooperam entre si e outras baseadas em fenômenos físicos como o *Simulated Annealing* [3]. Estes procedimentos são chamados de *meta-*

heurísticas e possuem como principal característica a existência, em seu interior, de ferramentas que possibilitam se esquivar de ótimos locais ainda distantes da solução ótima do problema, assim como uma flexibilidade responsável por uma melhor adaptação ao espaço de busca.

O Problema de Recobrimento de Rotas (PRR), foco do estudo deste trabalho, é um problema de otimização de elevada complexidade, ainda pouco explorado pela literatura afim. O objetivo deste trabalho é fazer um estudo teórico do PRR, onde propõe-se uma versão generalizada do problema, denominada Problema de Recobrimento de Rotas Generalizado (PRRG), uma nova formulação matemática, assim como a análise das regras existentes para a redução do grafo associado ao PRR e a proposta de um novo conjunto de regras para a redução do grafo associado ao PRRG. Propõe-se ainda metaheurísticas baseadas em conceitos do método GRASP para a solução aproximada do problema.

Esta dissertação apresenta-se da seguinte forma: no Capítulo 2, é descrito o Problema de Recobrimento de Rotas, é proposto uma generalização do PRR e é apresentado a literatura existente para este problema, assim como problemas similares e aplicações. No Capítulo 3, é apresentada duas das três formulações matemáticas existentes para o PRR e propõe-se uma nova formulação matemática introduzindo variáveis de fluxo. No Capítulo 4, apresenta-se uma análise das regras de redução existentes na literatura para o PRR, assim como propõe-se um conjunto de regras para a redução do grafo associado ao PRRG. No Capítulo 5, é feita uma breve descrição das metaheurísticas GRASP (*Greedy Randomized Adaptive Search Procedure*) e VNS (*Variable Neigborhood Search*), propõe-se procedimentos para a construção de uma solução inicial para o PRRG e para a busca local em um espaço solução associado ao PRRG, assim como as metaheurísticas formadas por estes procedimentos. No Capítulo

6, são apresentados os resultados computacionais realizados acerca das propostas deste trabalho e, finalizando, o Capítulo 7, apresenta as conclusões e as propostas para a continuação do trabalho.

Capítulo 2

O Problema de Recobrimento de Rotas

2.1 Descrição do Problema

Definido formalmente em 1981 por John Current [4] em sua tese de Doutorado, o Problema de Recobrimento de Rotas (PRR), referido na literatura por *Covering Tour Problem* (CTP), é uma generalização do Problema do Caixeiro Viajante (*Traveling Salesman Problem* - TSP) e possui diversas aplicações incluindo roteamento.

O PRR pode ser definido na estrutura de um grafo simétrico $G = (V \cup W, E)$, completo e não direcionado, onde os seguintes conjuntos representam:

- $V \cup W = \{1, ..., m\}$ o conjunto dos vértices, onde $V \cap W = \emptyset$;
- $E = \{(i, j) \mid i, j \in V \cup W, i \neq j\}$ o conjunto das arestas;
- V o conjunto dos vértices que **podem ser visitados**;
- $T \subseteq V$ o conjunto dos vértices que **devem ser visitados**;
- W o conjunto dos vértices que **devem ser cobertos**;

Considera-se também que:

- Um vértice de W é considerado coberto se a distância entre ele e pelo menos um vértice pertencente à rota, for menor ou igual a uma distância $d \geq 0$, onde d é um valor previamente estabelecido;
- O vértice 1 representa o vértice origem ($1 \in T$);
- A matriz simétrica de distâncias $C = (c_{ij})$, definida sob o conjunto de arestas E, satisfaz a desigualdade triangular.

Objetivo do PRR: O Problema de Recobrimento de Rotas consiste em determinar uma rota ou um ciclo Hamiltoniano de comprimento mínimo sob um subconjunto de V. Esta rota deve conter todos os vértices do subconjunto $T \subseteq V$ e deve cobrir cada vértice do conjunto W.

A Figura 2.1 ilustra uma instância para o PRR com 29 vértices, onde a área de cobertura de cada vértice de W e de T está representada pela área delimitada pela circunferência de raio d. Note que, para cada $j \in W$, qualquer vértice $i \in V$, tal que $c_{ij} \leq d$, pode ser utilizado na rota para cobrir j. As arestas do grafo não são ilustradas para facilitar a visualização.

O PRR é considerado NP-difícil, já que se reduz ao Problema do Caixeiro Viajante (PCV) quando $d = 0$, $W = \emptyset$ e $V = T$.

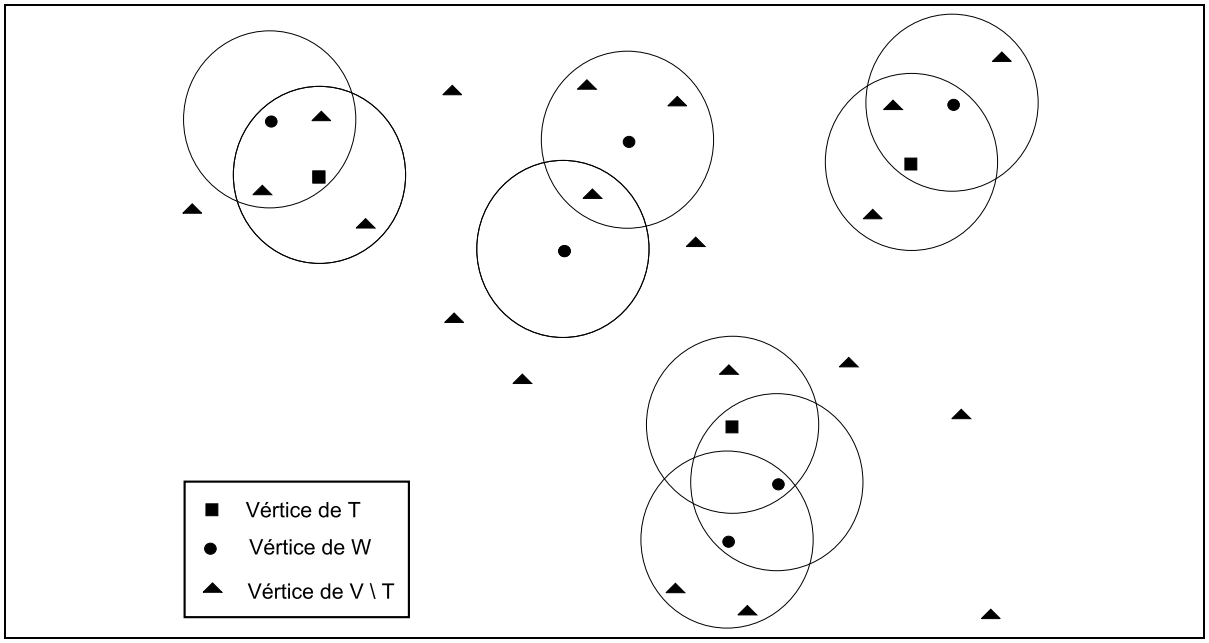

Figura 2.1: Grafo associado a uma instância do PRR.

2.2 Uma Generalização do Problema de Recobrimento de Rotas

De acordo com a descrição do PRR apresentada na seção 2.1, existe uma restrição que obriga a cobertura dos vértices do conjunto W. Portanto, para toda aplicação do PRR, sempre é necessário adotar uma distância de cobertura para garantir a existência de pelo menos um vértice de V na vizinhança de cada vértice do conjunto W. Com o intuito de abranger um número maior de aplicações, onde não exista a restrição de não incluir vértices de W na solução, esta seção apresenta uma generalização do PRR, aqui denominada Problema de Recobrimento de Rotas Genérico (PRRG).

O PRRG consiste em determinar uma rota de comprimento mínimo sob um subconjunto de vértices de $V \cup W$, permitindo que os vértices de W façam parte da rota solução cobrindo a si próprio e a outros vértices deste conjunto, quando for o

caso. Esta versão generalizada do PRR será abordada em todo este trabalho, sendo importante ressaltar que todos os procedimentos apresentados para o PRRG, podem facilmente ser adaptados para o PRR.

De acordo com a literatura [7], o PRR pode ser formulado como o Problema do Caixeiro Viajante Generalizado (PCVG), tornando possível particionar todos os vértices em um conjunto de grupamentos. Estendendo esta consideração ao PRRG, os grupamentos podem ser definidos do seguinte modo:

$$G_i = \{j \in V \backslash T \cup W \mid c_{ij} \leq d\}, \qquad \forall i \in W \tag{2.1}$$

$$G_t = \{t\}, \qquad \forall t \in T \tag{2.2}$$

Considerando a Figura 2.1, pode-se facilmente visualizar os grupamentos desta instância. Considere cada grupamento como o conjunto formado pelos vértices delimitados por cada vizinhança, isto é, todos os vértices pertencentes ao interior de cada circunferência. Note que, os grupamentos G_t, tal que $t \in T$, possuem cardinalidade igual a uma unidade, já que somente relacionam o próprio vértice que lhe dá origem. Note também que, por definção, os grupamentos associados aos vértices $i \in W$, não possuem nenhum vértice $j \in T$, mesmo este estando em sua vizinhança.

2.3 Literatura Existente

Apesar de apresentar grande aplicabilidade a problemas reais, o PRR não tem recebido muita atenção na literatura, desde que foi introduzido em 1981 por Current [4].

Current and Schilling propõem em [5] e [6] uma heurística para gerar um

conjunto de soluções para duas versões distintas do problema. A primeira versão do problema consiste em minimizar somente o comprimento da rota, enquanto a segunda, maximiza o número de vértices cobertos por ela.

Gendreau, Laporte e Semet [7], apresentam e analisam uma formulação matemática para o PRR (descrita na seção 3.1). Neste mesmo trabalho, é proposto um algoritmo exato do tipo *Branch and Cut* e uma heurística, cuja função é produzir um limite superior inicial para o algoritmo exato. Esta heurística combina a heurística GENIUS para o Problema do Caixeiro Viajante [8] com o algoritmo PRIMAL1 para o *Set Covering Problem* [9]. A heurística GENIUS para o TSP, consiste de duas fases. A primeira fase, chamada GENI (*Generalized Insertion*), inicia-se com a seleção arbitrária de três vértices, formando a rota parcial. A partir desta, um novo vértice é inserido a cada passo até que todos façam parte da rota. Uma diferença em relação as demais formas de inserção de vértices, é que neste método as inserções não necessariamente ocorrem entre dois vértices consecutivos da subrota. Após esta fase de inserção, inicia-se a fase de reotimização, conhecida por US (*Unstringing and Stringing*). Nesta reotimização, cada vértice é removido e reinserido na rota, usando o mesmo princípio da fase de insersão.

A outra heurística utilizada, PRIMAL1, inclui gradualmente na solução, variáveis (colunas) de acordo com um critério "guloso', visando a minimização de uma função $f(c_k, b_k)$, onde a cada passo, b_k representa o número de vértices $h \in W$, com $\delta_{hk} = 1$, ainda não cobertos pela rota (δ_{ij} é um coeficiente binário que é igual a 1, se e somente se, $i \in W$ pode ser coberto por $j \in V$). Os três critérios utilizados na heurística para o PRR foram sugeridos por Balas e Ho para o *Set Covering Problem* [9].

A heurística apresentada por Gendreau *et al.* em [7] constrói a rota

iterativamente inserindo os vértices na subrota (rota parcial) existente, utilizando a heurística GENIUS. A seleção dos vértices a serem inseridos a cada iteração é feita computando-se um custo c_k, igual ao menor custo de inserí-lo na subrota em estudo. O vértice é finalmente escolhido de acordo com o critério "guloso" definido, assim como na heurística PRIMAL1. A inserção prossegue até que todos os vértices de W estejam cobertos.

O algoritmo exato resolve um Problema de Programação Linear, contendo um subconjunto de restrições válidas, a partir de um vértice genérico da árvore. É feito uma separação de restrições violadas pela relaxação linear, introduzindo algumas destas restrições no problema corrente que é então reotimizado. Este processo é repetido até que uma solução viável seja obtida ou até que seja mais promissor particionar o espaço de busca, gerando duas novas soluções associadas.

Os testes são realizados com alguns problemas onde somente um subconjunto de vértices são visitados. Entre eles estão o *Prize Collecting Traveling Salesman Problem* (PCTSP) [10], o *Selective Traveling Salesman Problem* (STSP) [11] e o *Generalized Traveling Salesman Problem* (GTSP) [12]. Os algoritmos também são testados com uma série de problemas aleatórios, onde os $|V|+|W|$ vértices são gerados aleatoriamente em um quadrado de dimensões $[0,100]\times[0,100]$, considerando uma distribuição uniforme. Os conjuntos T e V são definidos considerando os $|T|$ e $|V|$ primeiros pontos respectivamente e W como o restante dos pontos. Os coeficientes c_{ij} (custo associado à aresta (i,j)) são computados como a distância Euclideana entre os vértices i e j. O valor da distância de cobertura é determinado por:

$$d = max(max_{k\in V\setminus T}\{\ min_{l\in W}\{c_{lk}\}, max_{l\in W}\{c_{l,k(l)}\}\}) \tag{2.3}$$

onde $k(l)$ é o índice do segundo vértice de $V\setminus T$ mais próximo de l. Desta forma, cada vértice de $V\setminus T$ cobre pelo menos um vértice de W e cada vértice de W é coberto por

pelo menos dois vértices de $V \backslash T$.

Outra regra também é utilizada para os vértices de W, onde estes não são expressos por coordenadas, mas listados em um vetor e cobertos por cada elemento de V com uma probabilidade p associada, considerada sucessivamente como 0.1, 0.2, 0.5 e 0.7. As instâncias são reduzidas de acordo com quatro critérios apresentados. Os algoritmos são codificados em C e executados em uma estação SunSPARC 1000. Testes são realizados com instâncias para $n = 50$, 75, 100, $|T| = 1$, $\lceil 0.25n \rceil$, $\lceil 0.50n \rceil$, $\lceil 0.75n \rceil$ e $|W| = n$, $2n$, $3n$, $4n$, $5n$. Para cada combinação desses parâmetros, 5 instâncias são resolvidas, envolvendo até 600 vértices (antes da redução do grafo). Aqueles envolvendo até 100 vértices, são resolvidas otimamente com um tempo computacional razoável. Os testes realizados constatam que a heurística obteve um bom desempenho para instâncias em que $|T|$ é bem pequeno. No algoritmo exato, o limite inferior inicial ficou bem perto da solução ótima (cerca de 0.5 por cento) e como conseqüência, a árvore de busca resultante é relativamente pequena. Entretanto, um certo esforço computacional é despendido em cada vértice e infelizmente os problemas testes usados pelos autores não se encontam disponíveis em bibliotecas públicas.

Em junho de 1999, Maniezzo *et al.* [13] apresentaram três algoritmos utilizando a metaheurística *Scatter Search* proposta por Glover [14]. A idéia básica desta metaheurística é a utilização de um conjunto R de soluções, chamado *Conjunto Referência*. Operadores de recombinação são aplicados nos pontos deste conjunto R, gerando novas soluções que posteriormente são utilizadas como pontos de partida para procedimentos de busca local. As melhores soluções obtidas ao final deste processo são inseridas no Conjunto Referência. Todo o procedimento é repetido por um determinado número de iterações e, geralmente, uma combinação linear é utilizada como operador de recombinação de soluções.

O primeiro algoritmo apresentado em [13], **SS-CTP1**, utiliza cortes para obter desigualdades que são posteriormente utilizadas para inicializar cada laço principal do algoritmo. O segundo, **SS-CTP2**, difere do primeiro por utilizar uma técnica diferente na geração de soluções, conhecida como *star paths*. O terceiro algoritmo, **SS-CTPB**, é elaborado de acordo com o algoritmo **SS1** apresentado em [15], diferindo dos dois anteriores por não usar cortes. Os três algoritmos apresentados em [13] compartilham uma série de procedimentos de problemas relacionados ao PRR e, por este motivo, um melhor detalhamento pode ser encontrado na referida literatura.

Quatro testes para a redução do grafo original, similares aos apresentados em [7], são utilizados. Estes testes são descritos e analizados posteriormente no Capítulo 4, seção 4.1. Os algoritmos são codificados em Fortran 77 e executados em uma *Silicon Graphics Indy* (MIPS R4400/processadores de 200Mhz), equipados com 256Mb de RAM em uma Irix 5.3. O CPLEX 4.0 [16] é utilizado para resolver o problema Lagrangeano. As instâncias são geradas como propostas por Gendreau et al. [7], descrito anteriormente. Os custos $\{c_{ij}\}$ são computados como valores inteiros iguais à $\lfloor e_{ij} + 0.5 \rfloor$, onde e_{ij} é a distância Euclideana entre os pontos i e j. Testes comparativos são realizados com a heurística proposta por Gendreau et al. [7], constatando que a heurística de Gendreau é muito mais rápida que o *Scatter Search*, proposto em [13], sendo em muitos casos, o problema é resolvido otimamente em menos tempo.

Não é de nosso conhecimento a existência de outro algoritmo, aproximado ou exato, proposto para o PRR.

2.4 Problemas Similares e Aplicações

Existem na literatura diversos problemas similares ao PRR. Entre eles estão: o *Shortest Covering Path Problem* (SCPP) [6, 17], o *Multi-Vehicle Covering Tour Problem* (m-CTP) [18], o *Median Tour Problem* (MTP), o *Maximal Covering Tour Problem* (MCTP) [19], entre outros.

O SCPP foi formulado em 1984 por Current *et al.* [17]. Definido sob um grafo completo e direcionado $G = (V, A)$, onde V é um conjunto de n vértices e A é um conjunto de m arcos (i, j), conectando o vértice i ao vértice j. O problema consiste em identificar um caminho de custo mínimo, partindo de um vértice origem s e visitando um vértice destino t ($s \neq t$), ambos previamente estabelecidos. O caminho deve cobrir todos os vértices do grafo. Um vértice é considerado coberto se distanciar no máximo um valor previamente estabelecido $d \geq 0$ de algum vértice do caminho. O *Shortest Covering Path Problem* reduz-se ao Problema de Recobrimento de Rotas se o vértice origem s coincidir com o vértice destino t, se o grafo G for não direcionado e se os vértices a serem cobertos não pertencerem ao caminho (solução do problema). Uma aplicação para o SCPP é o roteamento de despachos aéreos, onde os vértices (cidades) do caminho devem ser servidos por aeronaves. Os vértices cobertos por este caminho, mas não necessariamente contidos nele, devem ser cobertos por um meio de transporte secundário, geralmente terrestre. Neste exemplo, a distância máxima de cobertura, reflete uma distância ou um tempo máximo com a conexão ar-terra.

O m-CTP é definido sob um grafo $G = (V \cup W, E)$, completo e não direcionado, onde $V \cup W$ é o conjunto de vértices e $E = \{(i, j) : i, j \in V \cup W, i < j\}$ é o conjunto de arestas. O conjunto de vértices V, assim como no PRR, é particionado

em vértices que podem ser visitados ($V \backslash T$) e em vértices que devem ser visitados ($T \subseteq V$). O conjunto W contém os vértices que devem ser cobertos por uma distância $d \geq 0$ previamente estabelecida. A condição de cobertura de um vértice, assim como a matriz de distância $C = (c_{ij})$, são definidas como no problema tratado neste trabalho. Assim sendo, o m-CTP consiste em determinar um conjunto de m rotas de comprimento mínimo, de modo que as rotas satisfaçam:

1. Existem no máximo m rotas de comprimento mínimo, todas partindo e retornando ao depósito s ($s \in T$) previamente estabelecido;

2. Cada vértice de V pertence no máximo à uma rota, enquanto cada vértice de $T \subseteq V$ pertence à exatamente uma rota;

3. Cada vértice de W deve estar coberto por pelo menos uma rota, isto é, deve estar a uma distância máxima $d \geq 0$ de pelo menos um vértice pertencente a uma das m rotas (é assumido que o depósito s não cobre nenhum $i \in W$);

4. O número de vértices de cada rota (excluindo o depósito) não deve exceder um valor fixo p previamente estabelecido;

5. O comprimento de cada rota não deve exceder um valor fixo q previamente estabelecido.

O m-CTP se reduz ao PRR se $m = 1$, o valor pré-fixado p for grande o suficiente para conter, no pior caso, todos os vértices de V e q comportar o somatório do comprimento das arestas desta rota. Uma aplicação para o m-CTP pode ser associada ao problema de localização e roteamento de caixas de correios, considerando-se um subconjunto de locais candidatos [20], onde todos os usuários devem estar localizados a uma distância razoável de alguma das caixas. O custo da rota através de todas as caixas deve ser minimizado.

O *Prize Collecting Traveling Salesman Problem* (PCTSP) [21, 22] e o *Selective Traveling Salesman Problem* (STSP) [11] também podem ser vistos como problemas similares ao PRR. Ambos possuem um prêmio p_i, não negativo, associado a cada vértice i do grafo e consistem na construção de uma rota através de um subconjunto destes vértices. O PCTSP consiste na minimização do custo da rota, onde o total de prêmios coletados é no mínimo p. A cada vértice não pertencente à rota, existe uma penalidade associada a ser paga. O STSP consiste na maximização do total de prêmios coletados pela rota cujo comprimento total não deve exceder um certo valor r.

Pode-se ainda citar o *Median Tour Problem* (MTP) e o *Maximal Covering Tour Problem* (MCTP), que assemelham-se em muitos pontos. Em ambos problemas, a rota deve conter somente p dos n vértices do grafo. Adicionalmente, assim como buscam a minimização do comprimento total da rota, também objetivam a maximização do acesso à rota por parte dos vértices não pertencentes a ela. A diferença básica entre o MTP e o MCTP é que no primeiro deseja-se minimizar o custo total da distância de acesso à rota por parte dos vértices que não fazem parte dela, enquanto no segundo, deseja-se minimizar a demanda total dos vértices não cobertos pelos vértices da rota. Uma aplicação potencial do MTP e do MCTP, descrita por Oppong e Hodgson [23], é o roteamento de equipes para a entrega de medicamentos em países subdesenvolvidos, onde medicamentos somente podem ser entregues a um subconjunto de vilarejos. Todos os usuários dos demais vilarejos devem ser capazes de atingir pelo menos um vilarejo da rota, visando suprir a necessidade de serviços e medicamentos. O MTP pode ser usado para determinar que vilarejos visitar se o tempo médio de acesso é a restrição principal. Para o caso de existir uma distância máxima a ser percorrida pelos usuários, usa-se o MCTP.

Uma versão contínua do PRR é também chamada *Geometric Covering*

Salesman Problem (GCSP) [24], onde o conjunto de vértices não é um conjunto discreto, mas sim a região de um plano. Uma aplicação para o GCSP pode ser vista na determinação de rotas percorridas por robôs em polígonos cuja visibilidade é limitada por uma série de restrições [25] ou ainda na identificação aérea de focos de incêndio em florestas [26].

Existem ainda muitas outras aplicações para o problema em estudo. O roteamento de aeronaves em vôos noturnos, onde a rota não inclui a visita em todas as cidades diretamente e o planejamento de paradas de um circo durante uma estação são exemplos que podem ser mencionados. Este último é também conhecido como *Traveling Circus Problem* [27] e consiste em encontrar uma rota onde as localidades não incluídas no planejamento de paradas estejam acessíveis a pelo menos uma das incluídas, acrescentando para cada localidade não visitada, uma penalidade.

Capítulo 3

Formulações Matemáticas para o PRR

De nosso conhecimento, somente três formulações para o PRR são apresentadas na literatura. O problema foi primeiramente formulado em 1989 por Current e Schilling [5]. Mais tarde, Gendreau *et al.* [7] e Maniezzo *et al.* [13] apresentaram novas formulações. Este capítulo descreve duas das três formulações existentes e posteriormente, na seção 3.3, apresenta uma nova formulação, onde é proposto um novo conjunto de restrições para evitar ciclos desconexos do vértice origem.

3.1 Formulação de Gendreau, Laporte e Semet

Segundo Gendreau et al. [7], o PRR pode ser formulado como um Problema de Programação Linear Inteira (PPLI) da seguinte forma:

- Para todo $k \in V$, y_k é definido como uma variável binária igual a 1, se e somente se, o vértice k estiver na rota e zero, caso contrário;

- Para todo $k \in T$, y_k é necessariamente igual a 1;
- Para $i, j \in V$, com $i < j$, x_{ij} é definido como uma variável binária igual a 1, se e somente se, a aresta (i, j) pertencer à rota e zero, caso contrário;
- δ_{lk} é um coeficiente binário igual a 1, se e somente se, $l \in W$ pude ser coberto por $k \in V$;
- $S_l = \{k \in V \mid \delta_{lk} = 1\}$ é o conjunto dos vértices $k \in V$ que cobrem o vértice $l \in W$;
- S é qualquer subconjunto próprio de V, tal que $T \backslash S \neq \emptyset$;
- λ corresponde a $\{i \in S,\ j \in V \backslash S \text{ ou } j \in S,\ i \in V \backslash S\}$, ou seja, λ corresponde às arestas (i, j) que possuem uma de suas extremidades no conjunto S e a outra no conjunto $(V \backslash S)$;
- Por definição, a rota degenerada $\{1, 1\}$, onde 1 é o depósito, é considerada inviável.

De acordo com estas considerações, o PRR como um PPLI é descrito da seguinte forma:

$$(PRR)\ minimizar \sum_{i<j} c_{ij} x_{ij}, \tag{3.1}$$

sujeito à:

$$\sum_{k \in S_l} y_k \geq 1 \qquad (l \in W), \tag{3.2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2y_k \qquad (k \in V), \tag{3.3}$$

$$\sum_{\lambda} x_{ij} \geq 2y_t \quad (S \subset V,\ 2 \leq \mid S \mid \leq n-2,\ T \backslash S \neq \emptyset,\ t \in S), \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \qquad (1 \leq i < j \leq n), \tag{3.5}$$

$$y_k = 1 \qquad (k \in T), \tag{3.6}$$

$$y_k \in \{0,1\} \qquad (k \in V \backslash T). \tag{3.7}$$

Nesta formulação, as restrições (3.2) asseguram que todos os vértices de W estejam cobertos pela rota, (3.3) determinam que todos os vértices da rota possuam obrigatoriamente grau 2, isto é, para cada vértice da rota, somente duas arestas incidem nele. As restrições (3.4) são responsáveis pela conectividade da rota, forçando a presença de pelo menos duas arestas entre os conjuntos S e $V \backslash S$. Finalmente, as restrições (3.5), (3.6) e (3.7) referem-se às condições de integralidade do problema.

3.2 Formulação de Maniezzo, Baldacci, Boschetti e Zamboni

Baseada em um princípio distinto do utilizado na formulação descrita na seção 3.1, a formulação proposta por Maniezzo *et al.* [13] descreve o PRR como um Problema de Programação Linear Inteira, associando duas variáveis de fluxo x_{ij} e x_{ji} a cada aresta (i, j) do grafo. Este modelo foi proposto originalmente por Finke, Claus e Gunn [28] para o TSP. Algumas considerações são feitas:

- A rota parte de um vértice origem s, sendo s qualquer vértice do conjunto T;
- x_{ij} é a variável associada ao fluxo na aresta (i, j), $i, j \in V, i \neq j$. Se a rota vai do vértice i ao vértice j, então x_{ij} representa o número de vértices que podem ser visitados e x_{ji} representa o número de vértices já visitados. Desta forma, a variável x_{ij} define dois circuitos para qualquer rota que representar uma solução viável;

- ξ_{ij} é uma variável binária igual a 1, se e somente se, a aresta $(i,j) \in E$ está na solução e zero, caso contrário;

- y_i é uma variável binária igual a 1, se e somente se, o vértice $i \in V$ for visitado e zero, caso contrário.

Desta forma, o PRR pode ser descrito como um PPLI da seguinte forma:

$$(PRR)\ minimizar \sum_{(i,j)\in E} c_{ij}\xi_{ij}, \tag{3.8}$$

sujeito à:

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = -2y_i, \qquad (i \in V) \tag{3.9}$$

$$\sum_{j\in V} (x_{ij} + x_{ji}) = 2y_i(|V| - 1), \qquad (i \in V) \tag{3.10}$$

$$\sum_{j\in G_h} y_j \geq 1, \qquad (G_h = \{i \in V : c_{ih} \leq d, h \in W\}) \tag{3.11}$$

$$x_{ij} + x_{ji} = (n-1)\xi_{ij}, \qquad (i,j) \in E \tag{3.12}$$

$$x_{ij} \geq 0, \qquad i,j \in V, i \neq j \tag{3.13}$$

$$y_j = 1 \qquad (j \in T), \tag{3.14}$$

$$y_j \in \{0,1\}, \qquad (j \in V \backslash T). \tag{3.15}$$

$$\xi_{ij} \in \{0,1\}, \qquad (i,j) \in E \tag{3.16}$$

Nesta formulação, as restrições (3.9) e (3.13) definem um fluxo viável para as variáveis x_{ij}. As restrições (3.10) e (3.12) asseguram que todo vértice i pertencente à rota (isto é, $y_i = 1$) possui necessariamente grau 2, ou seja, exatamente duas arestas incidem nele. As restrições (3.11) garantem a cobertura de todos os vértices $h \in W$, garantindo que pelo menos um vértice $i \in V$, tal que $c_{ih} \leq d$ faça parte da rota. A presença de todos os vértices $i \in T$ na solução é assegurada pelas restrições (3.14). Finalmente, as restrições (3.15) e (3.16) definem as condições de integralidade do problema.

3.3 Uma Formulação para o PRRG

Esta seção propõe uma formulação de Programação Linear Inteira para o PRRG, que pode ser facilmente adaptada ao PRR. Nesta formulação, variáveis de fluxo são introduzidas com a finalidade de evitar a formação de ciclos desconexos da origem. A idéia é associar uma variável de fluxo à cada aresta (i, j) do grafo.

Considere:

- y_k, para $k \in V \cup W$, uma variável binária, igual a 1 se e somente se, o vértice k estiver na rota e zero, caso contrário;
- $s = 1$ é o vértice origem;
- Se $k \in T$, então y_k é necessariamente igual a 1;
- x_{ij} para todo $i, j \in V \cup W$ com $i \neq j$, é uma variável binária igual a 1, se e somente se, a aresta (i, j) pertence à rota e zero, caso contrário;
- $S_l = \{k \in V \cup W \mid c_{lk} \leq d,\ l \in W\ \}$, isto é, S_l é o conjunto de todos os vértices de $V \cup W$ que cobrem o vértice $l \in W$;
- Defina a variável de fluxo z_{ij} associada a cada $(i, j) \in E$, com $i \neq j$, como uma variável inteira não negativa, representando a quantidade de fluxo escoado no arco (i, j).

De posse destas considerações, a formulação proposta apresenta-se da seguinte forma:

$$(PRRG)\ minimize \sum_{i<j \,|\, i,j \in V \cup W} c_{ij} x_{ij} \tag{3.17}$$

sujeito à:

$$\sum_{k \in S_l} y_k \geq 1 \qquad (\forall l \in W), \tag{3.18}$$

$$\sum_{i<k} x_{ik} + \sum_{j<k} x_{kj} = 2y_k \qquad (\forall k \in V \cup W), \tag{3.19}$$

$$\sum_{j \in V \cup W} z_{kj} = \sum_{i \in V \cup W} z_{ik} + y_k \qquad (\forall k \in V \cup W - \{s\}), \tag{3.20}$$

$$\sum_{j \in V \cup W} z_{sj} = 1, \tag{3.21}$$

$$\sum_{j \in V \cup W} z_{js} = \sum_{j \in V \cup W} y_j, \tag{3.22}$$

$$x_{ij} \leq z_{ij} \quad (\forall i, \forall j \in V \cup W); \tag{3.23}$$

$$x_{ij} \geq (z_{ij})/(|V| + |W| + 1) \qquad (\forall i, \forall j \in V \cup W), \tag{3.24}$$

$$y_k = 1 \qquad (\forall k \in T), \tag{3.25}$$

$$y_k \in \{0, 1\} \quad (\forall k \in V \backslash T); \tag{3.26}$$

$$x_{ij} \in \{0, 1\} \quad (\forall i, \forall j \in V \cup W) \tag{3.27}$$

$$z_{ij} \in Z^+ \quad (\forall i, \forall j \in V \cup W). \tag{3.28}$$

Nesta formulação, definidas pelas condições (3.17)-(3.28), as restrições (3.18) asseguram que cada vértice de W é coberto por pelo menos um vértice da rota. As restrições (3.19) restringem a dois o grau de cada vértice da rota. As restrições (3.20), (3.21) e (3.22), asseguram a conectividade da rota. As restrições (3.23) e (3.24), asseguram que a rota gerada a partir da variável de fluxo (z_{ij}) coincide com a gerada pela variável de decisão (x_{ij}). As restrições (3.25) asseguram que cada vértice de T necessariamente faz parte da rota. Finalmente, as restrições (3.26), (3.27) e (3.28) representam as condições de integralidade do problema.

A proposição a seguir assegura que as restrições propostas (3.20), (3.21) e (3.22) realmente evitam a formação de rotas desconexas.

Proposição 3.3.1 *Seja S uma solução viável para o PRRG gerada pela formulação definida pelas restrições (3.17)-(3.28). Então S define um único ciclo contendo a origem s.*

Prova:

Suponha por absurdo que S seja viável, mas não contém a origem s. Então $S = \{1, 2, 3..., \alpha, 1\}$, tal que $\forall i \in S$, $i \neq s$. Assim, se

$$z_{12} = f \geq 0 \tag{3.29}$$

então, usando (3.20), temos:

$$z_{23} = f + 1 \tag{3.30}$$

Aplicando sucessivamente (3.20), temos que:

$$z_{\alpha 1} = f + (\alpha - 1), \qquad (\alpha > 1) \tag{3.31}$$

Desta forma, por (3.20) e (3.31), temos:

$$z_{12} = f + (\alpha - 1) + 1 = f + \alpha \qquad (\alpha > 1) \tag{3.32}$$

Logo, por (3.29) e (3.32),

$$f = z_{12} = f + \alpha \qquad (\alpha > 1) \tag{3.33}$$

O que é um absurdo. Portanto, se S satisfaz (3.20), então S necessariamente contém a origem s. ■

Capítulo 4

Regras de Redução

Os Problemas de Otimização consistem em alguma forma de busca, onde deseja-se encontrar em um conjunto, um ou mais componentes que atendam condições previamente estabelecidas. Para os problemas desta natureza, onde o objetivo consiste em encontrar um ótimo global, a capacidade da exploração do espaço de busca é essencial para a obtenção de uma solução de boa qualidade. Entretanto, na maioria das vezes, a qualidade da busca fica comprometida pela grande quantidade de informações que devem ser analisadas. Nos últimos anos, além do desenvolvimento de algoritmos cada vez mais eficientes, diversos procedimentos para a redução do espaço de busca têm sido incorporados aos algoritmos para a solução de problemas altamente combinatórios. O objetivo principal dos procedimentos de redução, é isolar um subconjunto de candidatos da região de busca, sem excluir nenhuma solução ótima. Este capítulo descreve e analisa regras para a redução do grafo associado ao PRR existentes na literatura e apresenta uma regra adicional para a redução do grafo associado ao PRRG.

4.1 Redução do Grafo associado ao PRR

Similar a muitos outros problemas da literatura, onde somente um subconjunto de vértices do grafo associado compõe uma solução viável, o PRR possui algumas regras para a redução do seu espaço de busca descritas na literatura. Gendreau, Laporte and Semet [7] introduziram quatro regras para a redução do PRR, utilizadas posteriormente por Maniezzo *et al.* [13]:

Considere W, T e V, conjunto dos vértices do grafo associado ao PRR como definidos anteriormente e ξ_{ij}, uma variável binária que representa a condição de cobertura do vértice i em relação ao vértice j, onde $\xi_{ij} = 1$ se o vértice i é coberto pelo vértice j e zero caso contrário. O conjunto das regras de redução apresentadas em [7] é descrito por:

1. Remover $i \in W$, se $\forall j \in V \backslash T$, $\xi_{ij} = 1$;
2. Remover $i \in W$, se existir $j \neq i$, com $j \in W$, tal que $\xi_{ik} \leq \xi_{jk}$, $\forall k \in V \backslash T$;
3. Remover $i \in W$, se existir $j \in T$, tal que j cubra i, ou seja, $\xi_{ij} = 1$;
4. Remover $i \in V \backslash T$, se $\forall j \in W$, $\xi_{ji} = 0$, ou seja, se i não cobre nenhum $j \in W$.

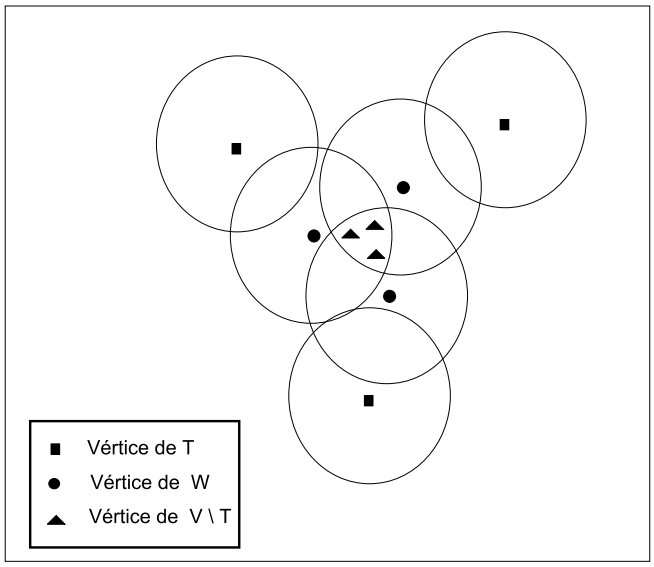

Figura 4.1: Grafo associado a uma instância do PRRG para análise da regra (1) da literatura.

4.1.1 Análise das Regras existentes para Redução do PRR associadas ao PRRG

Esta seção analisa cada uma das regras para a redução do grafo associado ao PRR existentes na literatura, verificando a sua aplicabilidade tanto para o PRR, quanto para o PRRG.

Considere o grafo de uma instância do PRR ilustrado na Figura 4.1. Neste grafo, os vértices $i \in W$ podem ser cobertos por qualquer vértice optativo $j \in V \backslash T$. Logo, uma solução viável para este caso pode ser obtida considerando uma rota que contenha todos os vértices de T mais qualquer vértice de $V \backslash T$.

Se a regra de redução **(1)** for aplicada ao grafo da Figura 4.1, todos os vértices de W são removidos (Figura 4.2). Com isso, uma solução viável para grafo reduzido basta conter os vértices obrigatórios (Figura 4.3), já que W torna-se um conjunto vazio e a presença dos vértices optativos ($V \backslash T$) na rota não se justifica, uma

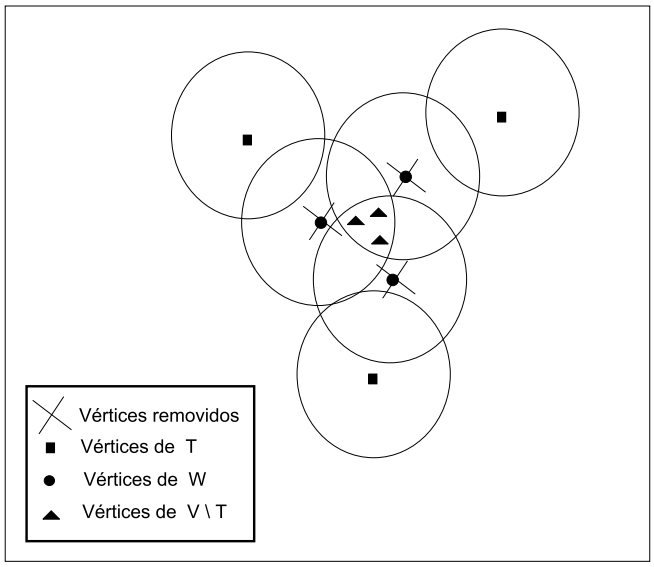

Figura 4.2: Redução do grafo da Figura 4.1 (regra 1).

vez que a métrica utilizada é a Euclideana. Logo, a maioria dos algoritmos encontrará uma solução viável associada ao grafo reduzido (Figura 4.3) que é inviável para o grafo original (Figura 4.1).

A regra **(2)** elimina do grafo original todo vértice de $i \in W$, quando existir um outro vértice $j \neq i$ em W, tal que todo $k \in V \backslash T$ pertencente à vizinhança de i, também é pertencente à vizinhança de j, mas nem todo $k \in V \backslash T$ pertencente à vizinhança de j, pertence à vizinhança de i. Observe esta situação ilustrada na Figura 4.4. Se a redução for realizada de acordo com a regra **(2)**, um vértice de W será eliminado (Figura 4.5) e o grafo reduzido terá como solução ótima (Figura 4.6), uma solução inviável ao grafo original. Para o grafo considerado, uma solução viável ao grafo original deve cobrir os dois vérices de W além de conter os três vértices de T (Figura 4.7).

A Figura 4.8 ilustra um grafo associado a uma instância do PRRG. Este grafo possui um vétice de W pertencente tanto a vizinhança de um vértice obrigatório (T) quanto de um outo vértice de W. Aplicando a regra **(3)**, que elimina do grafo

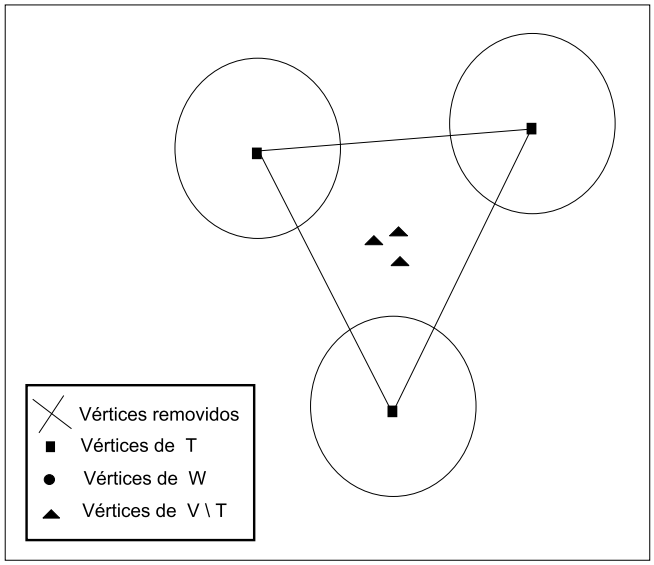

Figura 4.3: Grafo da Figura 4.1 após a redução, com a rota ótima associada.

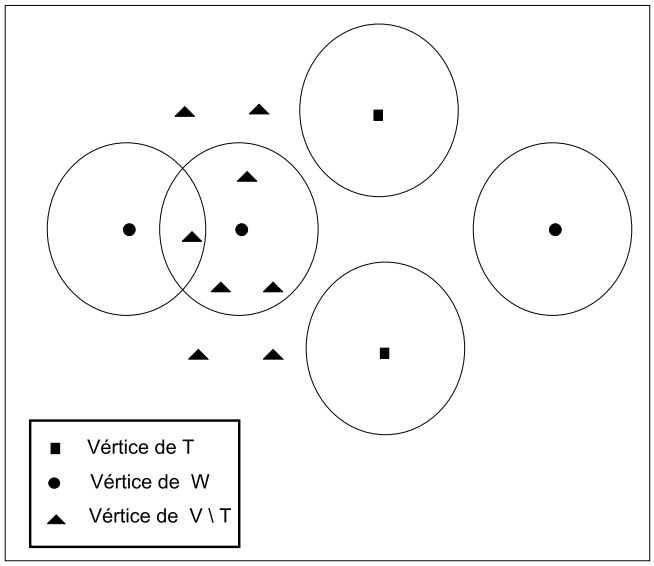

Figura 4.4: Grafo associado a uma instância do PRRG para análise da regra (2) da literatura.

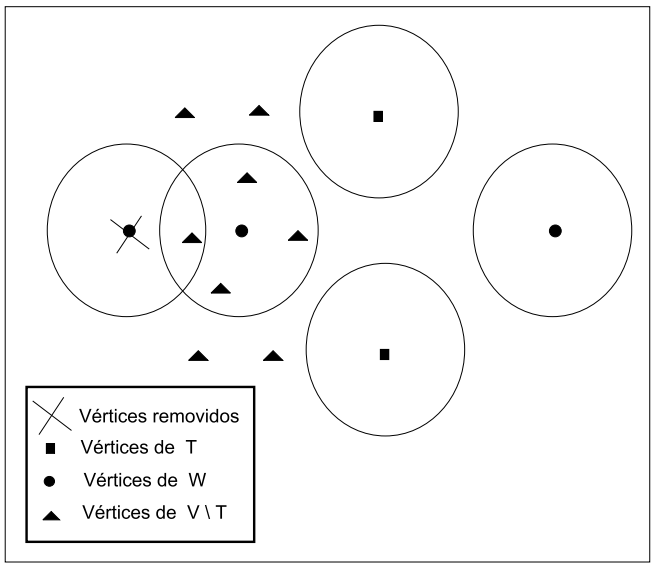

Figura 4.5: Redução do grafo da Figura 4.4 (regra 2)

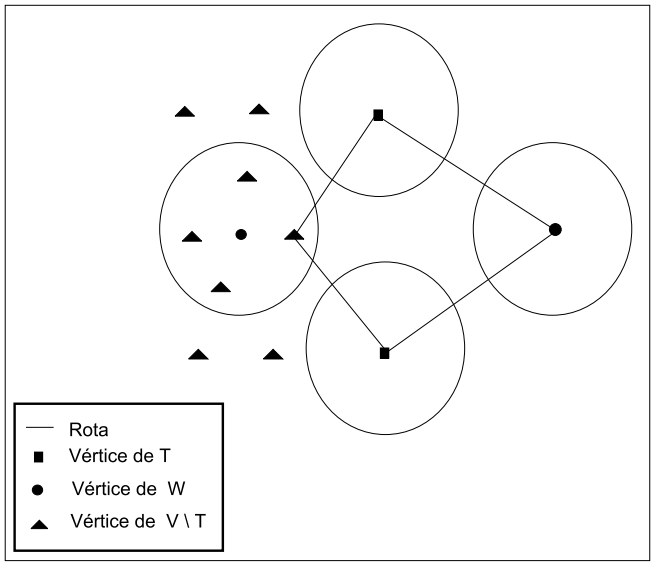

Figura 4.6: Grafo da Figura 4.4 após a redução, onde a rota ótima associada é viável somente no grafo reduzido.

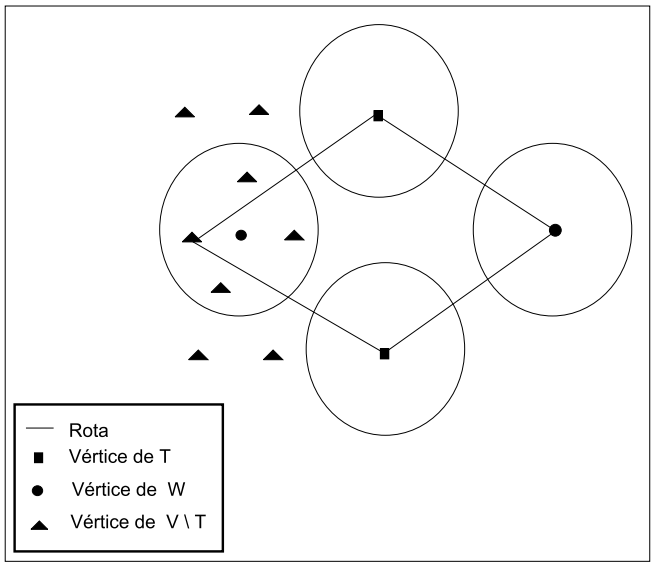

Figura 4.7: Grafo reduzido (regra 2) com a rota ótima associada ao grafo original.

original todo vértice de W coberto por pelo menos um vétice de T, um vértice do conjunto W é eliminado (Figura 4.9). Como todos os vértices do conjunto T fazem parte de qualquer solução viável, tanto no PRR quanto no PRRG, nenhum dos vértices eliminados pela regra **(3)** deixa de ser coberto, assegurando que toda solução viável no grafo reduzido é também viável no grafo original.

A Figura 4.10 ilustra a solução ótima do grafo reduzido. Contudo, apesar da solução ilustrada nesta figura ser a melhor associada ao grafo reduzido, não é a melhor associada ao grafo original (Figura 4.11. A solução ótima do grafo original foi eliminada do espaço de busca quando o vértice de W foi removido pela regra **(3)**. Portanto, apesar de não inviabilizar soluções do grafo reduzido quando associadas ao grafo original, assim como as regras **(1)** e **(2)**, a regra **(3)** não se aplica ao PRRG, pois elimina os vértices de W sem considerar a possibilidade de serem utilizados na cobertura de outros vértices do mesmo conjunto.

Nenhuma consideração importante foi observada na regra **(4)**, já que esta elimina somente os vértices optativos não pertencentes a nenhuma vizinhança de W,

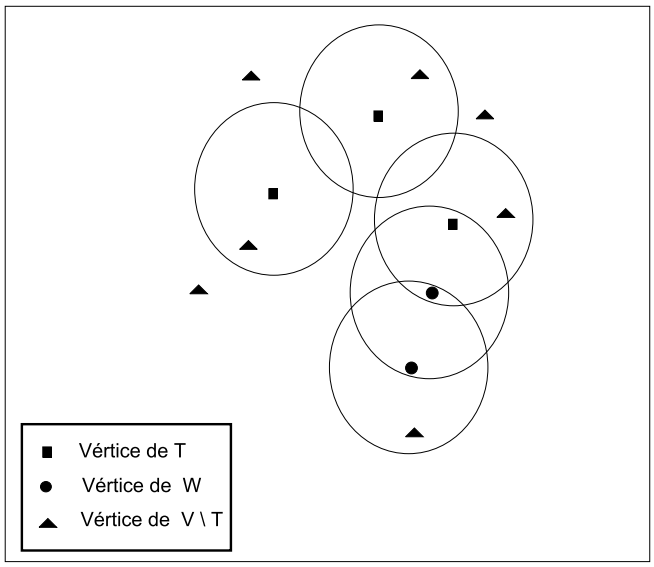

Figura 4.8: Grafo associado a uma instância do PRRG para análise da regra (3) da literatura.

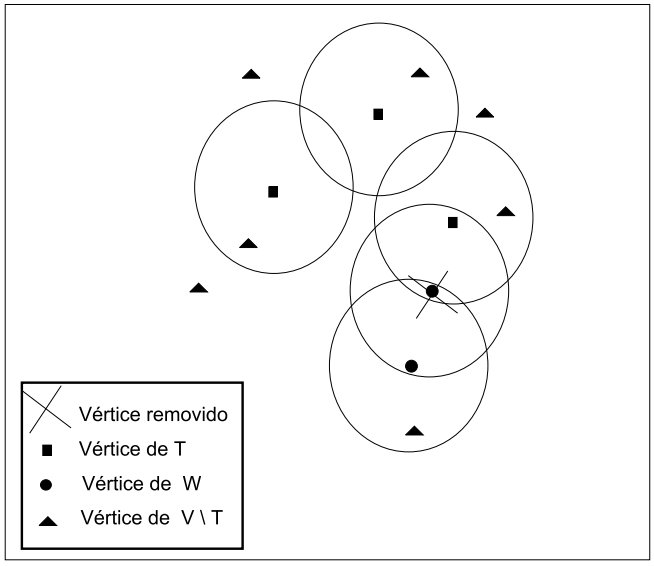

Figura 4.9: Redução do grafo da Figura 4.8 (regra 3)

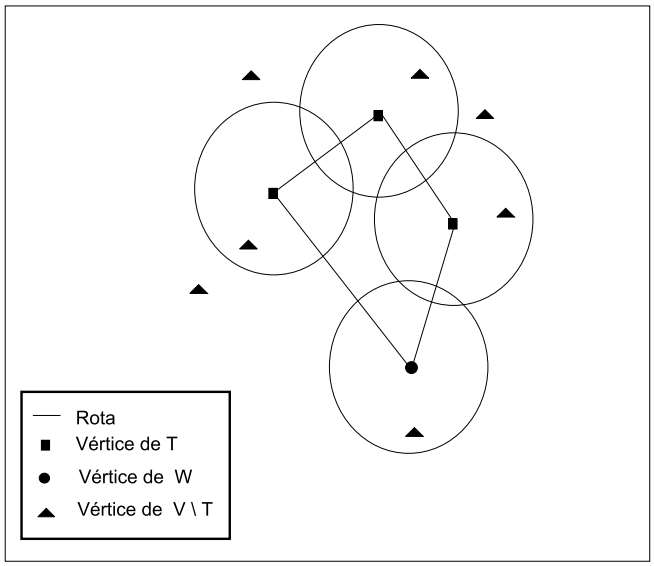

Figura 4.10: Grafo da Figura 4.8 após a redução com a rota ótima associada.

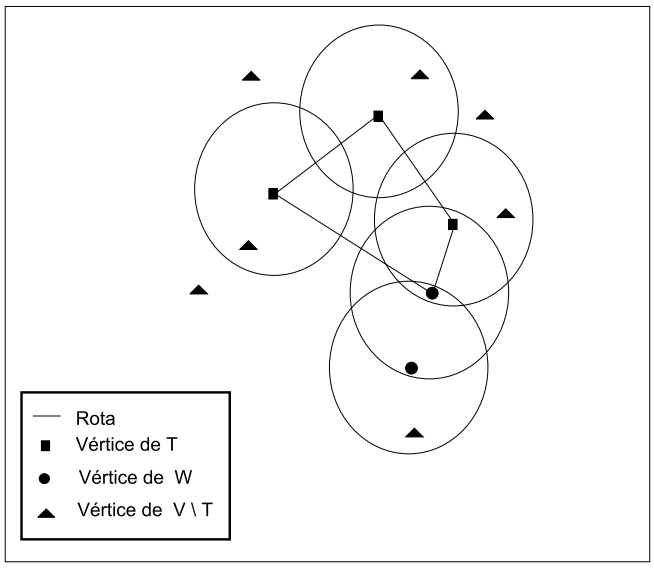

Figura 4.11: Grafo original da Figura 4.8 com a rota ótima associada.

ou seja, que não podem ser utilizados na cobertura de nenhum vértices deste conjunto.

4.2 Redução do Grafo associado ao PRRG

Esta seção apresenta um conjunto de regras para a redução do grafo associado à uma instância do PRRG. Este conjunto é composto por uma regra desenvolvida neste trabalho, associada a outra da literatura.

A grande diferença existente na redução do espaço de soluções do PRR e do PRRG está na remoção dos vértices do conjunto W. Remover um vértice do conjunto W no PRRG, consiste em antes verificar não somente a sua condição de cobertura, mas também verificar a condição de cobertura dos vértice da sua vizinhança. Ou seja, é necessário analisar que tipo de contribuição ele pode proporcionar aos componentes da sua vizinhança.

Seja uma matriz $Cob = (cob_{ij})$, $|W| \times |V \cup W|$, de zeros e uns $(0-1)$, tal que $cob_{ij} = 1$, $\forall i \in W, j \in V \cup W$, se e somente se, $c_{ij} \leq d$. Sendo assim, as regras para a redução do grafo associado ao PRRG são apresentadas da seguinte forma:

- **R1** - Transformar em um vértice optativo $l \in V \backslash T$, todos os vértices $i \in W$ tal que $cob_{ji} = 1$ e $cob_{ik} = 1$, com $j \in W$ e $k \in T$;
- **R2** - Remover de $V \backslash T$ todos os vértices i tal que $cob_{ji} = 0$, $\forall j \in W$.

A idéia básica da regra proposta **(R1)** é primeiramente verificar que vértices de W estão sendo cobertos por algum vértice obrigatório de T. São estes os vértices analisados pela regra **(R1)**. No PRRG, os vértices de W podem ser utilizados

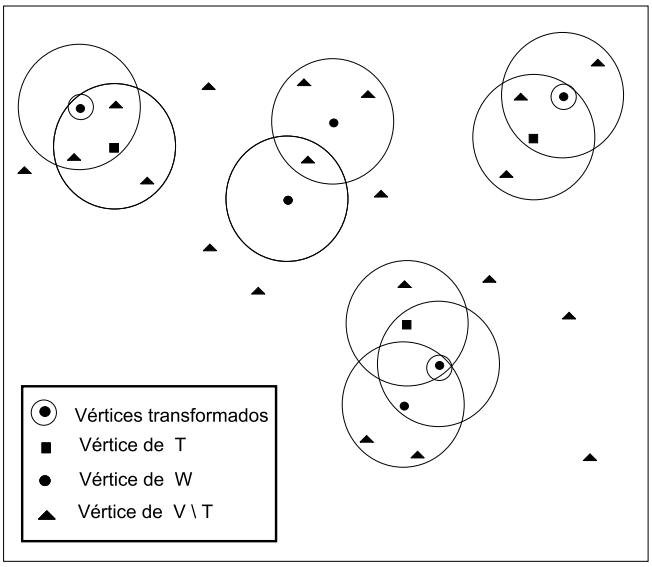

Figura 4.12: Grafo da Figura 2.1 analisado pela regra (R1).

na solução do problema. Portanto, o simples fato de estarem sendo cobertos por nós obrigatoriamente na solução, não significa que eles podem ser removidos do grafo, pois ainda podem ser utilizados como vértices optativos de $V \backslash T$. Considere o grafo ilustrado na Figura 2.1. O grafo resultante do uso da regra **(R1)** é ilustrado na Figura 4.12.

Note que, a regra **(R1)** não elimina nenhum vértice. Ela apenas qualifica os vértices de W já cobertos, identificando os possíveis candidatos para serem removidos pela regra **(R2)**.

A regra **(R2)**, em conjunto com à regra proposta **(R1)**, reduz efetivamente o espaço de busca associado ao PRRG. A Figura 4.13 ilustra os vértices que são removidos do grafo original ao final da regra **(R2)**.

No exemplo da Figura 2.1, o grafo original apresentou uma redução de $58,6\%$, decorrente remoção de 2 vértices de W (tranformados em vértices de $V \backslash T$) mais 15 vértices optativos. A Figura 4.14 ilustra o grafo reduzido final.

Note que nenhum vértice transformado pela regra **(R1)**, que pode ser utilizado na cobertura de outros, é removido do grafo. Isto assegura que nenhuma

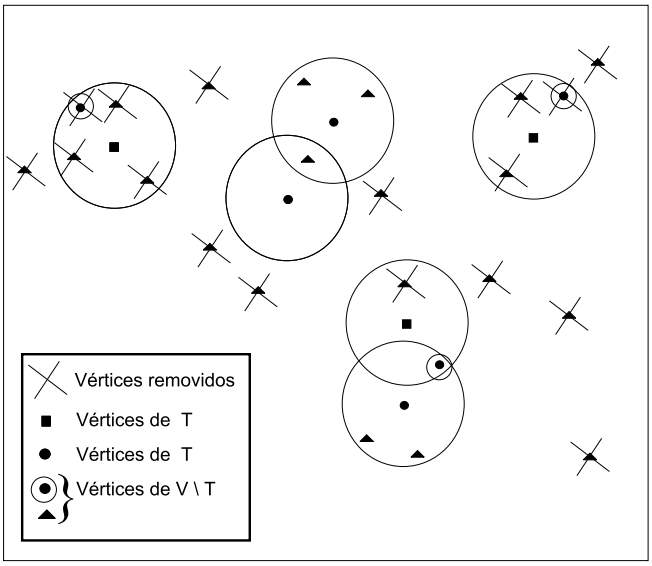

Figura 4.13: Grafo da Figura 2.1 analisado pela regra (R2).

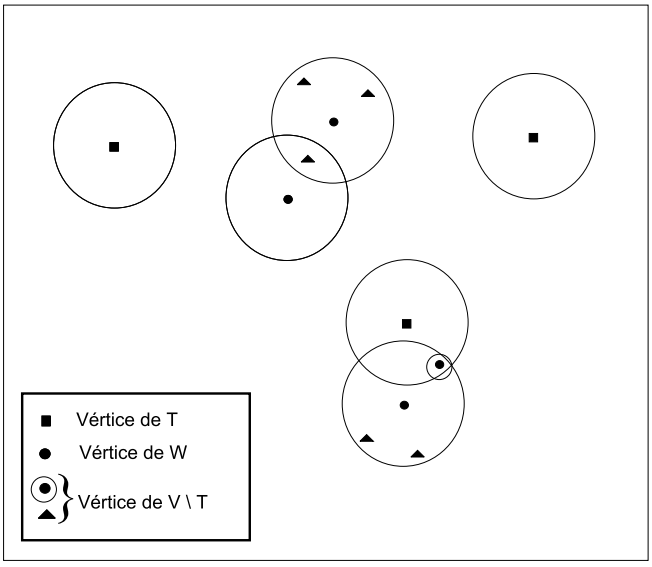

Figura 4.14: Grafo da Figura 2.1 reduzido.

solução que possa contribuir com um ganho no custo final da rota é eliminada do espaço de busca pelo conjunto de regras apresentadas neste trabalho. Além disto, todo vértice do conjunto W que não é removido deixa de ser analisado quanto à sua restrição de cobertura, acarretando um ganho no desempenho de qualquer tipo de busca que possa ser realizada visando a obtenção de uma solução viável para o problema.

Capítulo 5

Metaheurísticas para o PRRG

Uma maneira de resolver problemas de Otimização Combinatória é listar todas as possíveis soluções e selecionar a melhor. Entretanto, apesar de teoricamente viável, a enumeração exaustiva não funciona na prática na maioria dos casos devido ao elevado número de soluções associadas aos problemas desta natureza. Nas últimas décadas, a área de Otimização tem assistido a um significativo desenvolvimento em termos de propostas para modelagem e obtenção de soluções exatas relacionadas à problemas com alto grau de complexidade. Vários algoritmos exatos foram desenvolvidos, aumentando com isso, a viabilidade o seu uso na prática. Em contrapartida, principalmente para os problemas de larga escala, estes algoritmos ainda não são capazes em muitos casos de encontrar uma solução ótima em um tempo computacional razoável. Uma saída para esta limitação tem sido o uso de técnicas heurísticas, cujo objetivo baseia-se na obtenção de uma solução de boa qualidade, geralmente um ótimo local próximo de uma solução ótima global, em um tempo computacional suportável. Contudo, as heurísticas convencionais têm os seus limitantes e o principal deles é a dificuldade de escapar de ótimos locais ainda distantes de um ótimo global. Mais recentemente, a partir da década de 1980, começaram a surgir heurísticas genéricas, conhecidas

como Metaheurísticas ou Heurísticas Inteligentes, cujas características principais são a utilização de ferramentas que possibilitam escapar de ótimos locais ainda distantes de uma solução ótima global, além de possuirem uma flexibilidade, responsável por uma melhor adaptação ao espaço de busca.

A grande quantidade de problemas de otimização de elevada complexidade encontrados em diferentes áreas, tem incentivado a pesquisa em metaherísticas eficientes, capazes de lidar com problemas cada vez mais complexos e dinâmicos. Algumas metaheurísticas são usualmente o resultado da adaptação de idéias de uma variedade de áreas, como por exemplo os Algoritmos Genéticos [1], baseados em fenômenos da Biologia, o *Simulated Annealing* [29], baseado em fenômenos físicos, as Redes Neurais [30], baseado na conexão neuronal e outros. Nos últimos anos, algumas metaheurísticas têm se destacado na solução de problemas altamente combinatórios pelos bons resultados computacionais obtidos, entre elas a Busca Tabu (BT) [31], o VNS (*Variable Neighborhood Search*) ([32], [33]) e o GRASP (*Greedy Randomized Adaptive Search Procedure*) [34]. Dentre estas três metodologias, a BT é a mais conhecida, com centenas de artigos disponíveis envolvendo variadas aplicações e propostas de diferentes algoritmos sequenciais e paralelos. Por outro lado, o GRASP e o VNS são técnicas bem mais recentes e por isso ainda não tão explorados pela literatura afim. Este fato, conjugado ao bom desempenho que ambos têm demonstrado em diversas aplicações, foi um dos incentivos para propor neste trabalho, o desenvolvimento de novas técnicas baseadas em GRASP e VNS para a solução aproximada do PRRG.

Este capítulo está estruturado da seguinte forma: a seção 5.1 apresenta uma breve descrição das características básicas das metaheurísticas GRASP e VNS; a seção 5.2 descreve os algoritmos de construção e de busca local propostos e finalmente, a seção 5.2.3 mostra as metaheurísticas propostas para o PRRG.

5.1 Metaheurísticas GRASP e VNS

5.1.1 Metaheurística GRASP

O método GRASP (*Greedy Randomized Adaptive Search Procedure*) foi desenvolvido em 1995 por Tom Feo e Maurício Resende [34] e consiste em um processo iterativo onde cada iteração possui duas fases: uma fase de construção, na qual uma solução viável é construída, seguida de uma fase de busca local.

A fase de construção é também iterativa e adaptativa, podendo ser "gulosa". Ela é iterativa porque uma solução inicial é construída elemento a elemento iterativamente. É adaptativa porque os benefícios associados a adição de cada elemento são levados de uma iteração para outra nesta fase. A componente "gulosa" está relacionada à seleção de cada elemento em uma lista de candidatos, conhecida na literatura como *Restricted Candidate List ou Lista de Candidatos Restrita* (LCR). O tamanho da LCR é determinado por um parâmetro α (linha 6 da Figura 5.1) que controla a aleatoriedade do algoritmo. Quando $\alpha = 0$, o procedimento de construção torna-se totalmente "guloso", pois se restringe a lista de candidatos a apenas um elemento. Quando $\alpha = 1$, a construção torna-se totalmente aleatória, já que todas as opções remanescentes são candidatas para a escolha do próximo elemento de uma solução. A função "gulosa" $g : C \longrightarrow \Re$ (linha 6 da Figura 5.1), mede o benefício de se selecionar cada elemento da LCR. A Figura 5.1 ilustra o procedimento básico para a fase de construção da metaheurística GRASP.

Algoritmo: 1 : *GRASP - Fase de Construção*

1 $x = \emptyset$;
2 Inicializar a lista de candidatos C;
3 **Enquanto** $C \neq \emptyset$ **faça**
4 $s_i = min\{g(t) \mid t \in C\}$;
5 $s_s = max\{g(t) \mid t \in C\}$;
6 LCR=$\{x \in C \mid g(x) \leq s_i + \alpha(s_s - s_i)\}$;
7 Selecionar s de LCR aleatoriamente;
8 $x = x \cup s$;
fim enquanto

Figura 5.1: Fase de Construção GRASP.

A fase de busca local consiste tipicamente em um procedimento de refinamento, já que a solução gerada na fase de construção do GRASP não necessariamente representa um ótimo local. De modo geral, é sempre benéfica a aplicação de uma busca local visando a melhoria da qualidade das soluções obtidas na fase de construção de uma heurística. Resumidamente, um algoritmo de busca local consiste na substituição da solução atual pela melhor solução de sua vizinhança. Na maioria dos casos, a busca se encerra quando nenhuma solução de melhor qualidade é encontrada na vizinhança. Os principais aspectos responsáveis pelo sucesso de uma busca local consistem na escolha eficiente da estrutura da vizinhança, na eficiência da técnica de busca empregada nesta vizinhança e também da solução inicial fornecida para esta busca. No Algoritmo GRASP estes módulos de construção e busca local são executados n vezes, onde n é um parâmetro de entrada. A realização de múltiplas iterações do GRASP pode ser interpretada como uma seleção estratégica do espaço de soluções. A Figura 5.2 ilustra o procedimento básico da metaheurística GRASP.

É muito difícil analisar formalmente a qualidade das soluções obtidas com a utilização da metodologia GRASP. Uma justificativa intuitiva é que a cada

Algoritmo: 2 : *GRASP Básico*

1 $f(x^*) = \infty$;
2 **para** k **de** 1 **até** n **faça**
3 Aplicar o procedimento de construção para obter uma solução x;
4 Aplicar busca local em x gerando uma nova solução x';
5 **se** $f(x') < f(x^*)$ **então**
6 $x^* = x'$;
fim se
fim para

Figura 5.2: Algoritmo GRASP Básico.

iteração do GRASP é produzida uma solução a partir de uma distribuição desconhecida de todos os resultados obtidos até então. A característica e a variância desta distribuição é proveniente da natureza da lista de candidatos restrita determinada pela função g. Se a lista de candidatos restrita contiver apenas um elemento, então apenas uma solução é produzida, acarretando em uma variância nula para a distribuição. Por outro lado, se o limite da cardinalidade for menos restritivo, diferentes soluções podem ser produzidas, implicando em uma variância maior. Neste caso, dependendo da função "gulosa", a qualidade da solução pode melhorar bastante, aumentando a probabilidade de se encontrar uma solução ótima, mesmo sendo necessário dispender um tempo computacional maior, já que o aumento da variância no GRASP implica na redução do valor esperado.

5.1.2 Metaheurística VNS

Proposta por Nenad Mladenović em 1995 [32, 33], a metaheurística VNS (*Variable Neigborhood Search*) baseia-se em uma sistemática troca de estruturas de vizinhanças associada a um algoritmo de busca local. Diferente de outras metaheurísticas baseadas em métodos de busca local, o VNS não segue uma trajetória, mas explora incrementalmente vizinhanças distantes da solução atual. Neste caso, características freqüentes da solução atual são guardadas e utilizadas para a obtenção de soluções vizinhas promissoras.

A principal característica do VNS é o uso de um conjunto finito N_k, $(k = 1, ..., k_{max})$, de estruturas de vizinhanças previamente selecionadas. A maioria das heurísticas de busca local utiliza apenas uma estrutura de vizinhança, podendo com isso, limitar a qualidade da solução final obtida caso a vizinhança seja inadequada.

O algoritmo básico da metaheurística VNS é ilustrado na Figura 5.3.

A geração aleatória da solução x' (linha 6 da Figura 5.3) é realizada no intuito de reduzir o risco de ciclagem, o que provavelmente ocorreria se alguma regra determinística fosse utilizada nesta etapa.

5.2 Procedimentos Propostos

O algoritmo GRASP necessita basicamente de um procedimento de construção e de um de busca local. Desta forma, apresenta-se a seguir, alternativas para a

Algoritmo Básico: 3 : *VNS - Variable Neighborhood Search*

1 *Inicialização:* Selecionar o conjunto de estruturas de vizinhanças N_k; encontrar uma solução inicial x; escolher um critério de parada;
2 $x^* = x$;
3 **enquanto** critério de parada não satisfeito **faça**
4 $k = 1$
5 **enquanto** $k < k_{max}$ **faça**
6 Gerar aleatoriamente uma solução x' na vizinhança N_k de x;
7 Aplicar um procedimento de busca local em x' para obter x^*;
8 **se** $f(x') < f(x^*)$ **então**
9 $x^* = x'$;
10 $k = 1$;
  senão
11 $k = k + 1$
  fim se
  fim enquanto
fim enquanto

Figura 5.3: Algoritmo VNS Básico.

etapas de construção e busca local desenvolvidas para comporem versões distintas de algoritmos GRASP. A princípio, estes procedimentos foram desenvolvidos para a solução do PRR e PRRG, mas podem ser adaptados para outros problemas de otimização cuja solução incorpora apenas parte do conjunto dos vértices do grafo associado.

5.2.1 Algoritmos de Construção

Um fator importante a ser considerado, além da eficiência do algoritmo de busca, é a qualidade da solução inicial. Esta seção apresenta três algoritmos para a construção de uma solução inicial viável para o PRRG. Cada um destes algoritmos possui características distintas, o que permite avaliar empiricamente quais dessas características melhor se adaptam ao problema tratado neste trabalho.

Algoritmo SoluçãoInicial_ADD

Este algoritmo baseia-se em um critério de inserção de vértices, onde uma solução parcial começa a ser construída a partir de uma rota contendo dois vértices do conjunto T (conjunto dos vértices obrigatórios). Partindo de $LDisp$, lista que contém todos os vértices do grafo não pertencentes à rota parcial, ou seja, disponíveis para a inserção , um vértice $k \in LDisp$ é selecionado como o próximo a ser inserido na rota. Esta seleção é realizada de acordo com a ordem decrescente do benefício de se realizar a inserção deste vértice na rota parcial. Este benefício, aqui também chamado de economia, é determinado pela função ρ dada por:

$$\rho(k, i, j) = [c_{ij} - (c_{ik} + c_{kj})] + (M \times \phi_k) \tag{5.1}$$

onde:

- $\rho(k, i, j)$ fornece o benefício (saving) de inserir o vértice k entre os vértices adjacentes i e j da rota corrente;

- k representa o índice do vértice candidato à inserção;

- i e j são os índices dos vértices adjacentes na rota corrente entre os quais o vértice k será inserido;

- c_{ij}, c_{ik} e c_{kj} são as distâncias Euclideanas entre os respectivos vértices ;

- M é um valor inteiro, definido como 5000, que representa a penalidade associada aos vértices de T não pertencentes à rota e aos vértices de W não cobertos por ela;

- ϕ é o número de vértices do conjunto $T \cup W$ cobertos por k ainda não cobertos pela rota parcial. Neste caso, um vértice $i \in T$ é considerado coberto por k se $i = k$.

Vértices são inseridos na rota parcial até que nenhuma economia positiva possa ser alcançada. Note que, de acordo com a função economia ρ, os vértices são inseridos enquanto uma solução viável não for encontrada, pois como o valor de M é definido maior que qualquer distância da matriz $C = (c_{ij})$, é sempre uma economia positiva no custo total da rota inserir novos vértices até que a viabilidade da solução seja atingida. O algoritmo para o procedimento descrito nesta seção é ilustrado na Figura 5.4.

Na linha 1 do algoritmo 4, obtém-se a rota parcial inicial formada pela escolha aleatória de dois vértices distintos do conjunto T. Mesmo que esta rota apresente um custo associado muito ruim, as inserções posteriores são capazes de ajustar este custo. Na linha 2, inicializa-se o custo da rota com um valor equivalente a uma multa associada a cada vértice de W não coberto (determinado por Num_W, onde $|W| \leq Num_W \leq 0$) e a cada vértice obrigatório não pertencente à rota inicial (Num_T, onde $|T| \leq Num_T \leq 0$). O valor de $Multa$ é definido em função da maior

Algoritmo 4 : *SoluçãoInicial_ADD*

1 $Rota = \{i - j - i\}$
2 $Custo\_rota = (Num_W + Num\_T - 2) \times Multa$
3 $LDisp = (V - \{i, j\}) \cup W$
4 **enquanto** existir economia na adição de um vértice **faça**
5 $escolhido$ = vértice de $LDisp$ que oferece a maior economia;
6 $LDisp = LDisp - \{escolhido\}$
7 $posicao$ = melhor posição (i, j) para inserir $escolhido$ na rota;
8 $Rota = Rota + (escolhido, posicao)$;
9 $Custo\_rota = Custo\_rota - \rho(escolhido, posicao)$;
fim enquanto

Figura 5.4: Algoritmo *SoluçãoInicial_ADD*.

distância possível entre quaisquer dois vértices do grafo, visando garantir a viabilidade das soluções geradas pelo algoritmo, já que a viabilidade está associada à cobertura de todo o conjunto W e à pertinência de todos os vértices obrigatórios. Como o valor de $Multa$ é definido maior que qualquer distância da matriz $C = (c_{ij})$, torna-se sempre uma economia positiva no custo total da rota inserir novos vértices até que a viabilidade da solução seja atingida. Na linha 3, é inicializada a lista de vértices disponíveis para a adição. Dela fazem parte todos os vértices do grafo não pertencentes à $Rota$. O algoritmo prossegue com a escolha de um vértice da lista de disponíveis $LDisp$ até que nenhuma economia possa ser atingida (linha 4). A variável $posicao$ armazena o par ordenado entre o qual $escolhido$ será inserido.

A Figura 5.5 ilustra, passo-a-passo, o algoritmo *SoluçãoInicial_ADD* utilizando a instância PRRG representada pelo grafo da Figura 4.14.

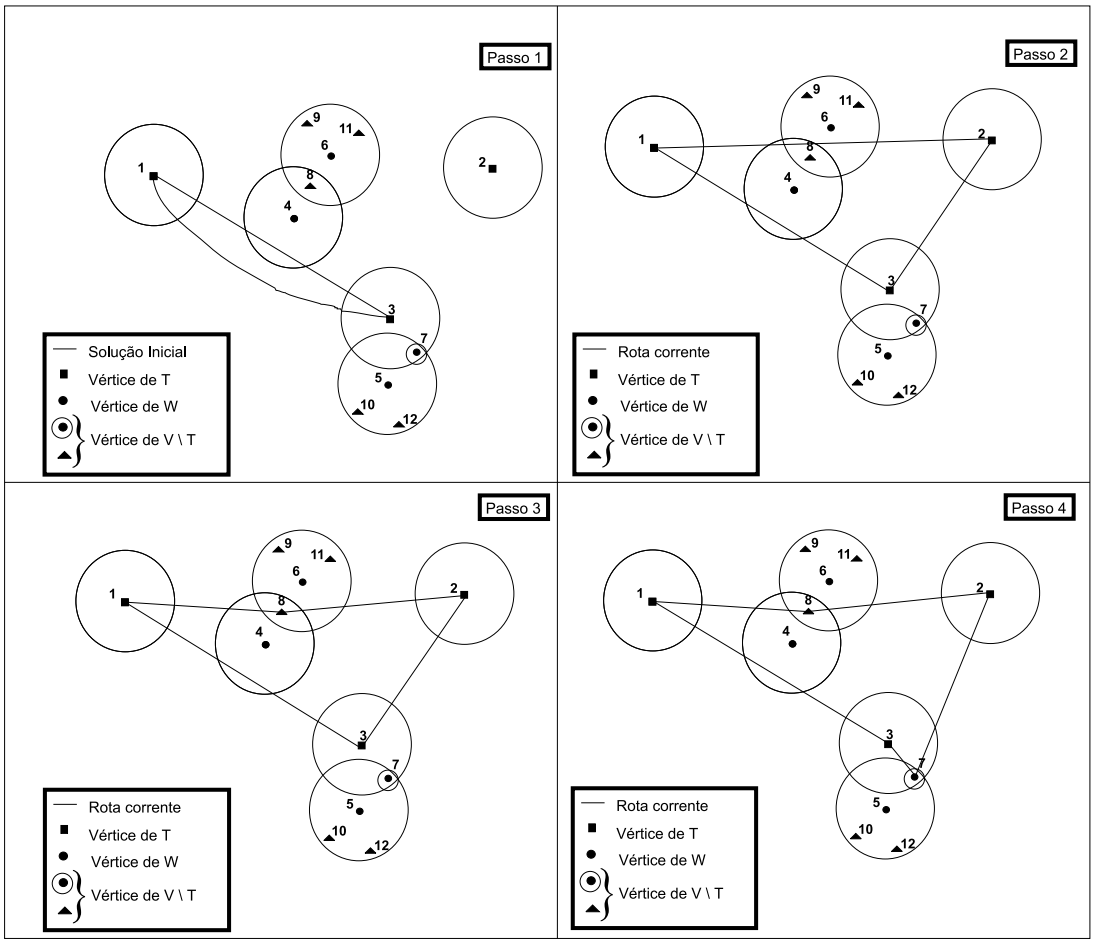

Figura 5.5: Passos de uma execução do algoritmo SoluçãoInicial_ADD

Algoritmo SoluçãoInicial_DROP

Este procedimento obtém uma solução inicial para o PRRG utilizando um critério de remoção de vértices. O ponto de partida deste algoritmo é fornecido por uma rota associada à uma solução do PCV, isto é, uma solução contendo todos os vértices do grafo. Esta é obtida com a utilização do algoritmo 4, diferindo apenas o critério de parada, onde os nós são inseridos até que $LDisp = \emptyset$, ou seja, até que todos os vértices já façam parte da rota. De posse da solução inicial do PCV, um processo de remoções sucessivas é então realizado. A seleção dos vértices a serem removidos é feita a partir de $LDisp$ (que neste caso contém todos os vértices da rota corrente), associando a cada elemento desta lista, a sua contribuição (economia) caso seja removido da solução. Este valor é determinado pela função γ, definida por:

$$\gamma(k) = [(c_{ik} + c_{kj}) - c_{ij}] - (M \times \phi_k) \tag{5.2}$$

onde:

- γ fornece o benefício (ou economia) de remover o vértice k da rota parcial;

- k é o índice do vértice candidato à remoção;

- i e j são, respectivamente, os índices dos vértices adjacentes ao vértice k;

- c_{ij}, c_{ik} e c_{kj} são as distâncias Euclideanas entre os respectivos vértices fornecidas pela matriz de distâncias;

- M é um valor inteiro, definido como 5000, que representa a penalidade associada aos vértices de T não pertencentes à rota e aos vértices de W não cobertos por ela;

- ϕ é o número de vértices do conjunto $T \cup W$ cobertos por k ainda não cobertos pela rota corrente. Neste caso, um vértice $i \in T$ é considerado coberto por k se $i = k$.

Os vértices são selecionados para a remoção pela ordem decrescente da

Algoritmo 5 : *SoluçãoInicial_DROP*

1 $Rota$ = rota com todos os vértices obtida pelo algoritmo 4
2 $Custo\_rota$ = custo associado à $Rota$ fornecido pelo algoritmo 4
3 $LDisp = (V \backslash T) \cup W$
4 **enquanto** existir economia na remoção de um vértice **faça**
5 $escolhido$ = vértice de $LDisp$ que oferece a maior economia;
6 $LDisp = LDisp - \{escolhido\}$;
7 $Rota = Rota - \{escolhido\}$;
8 $Custo\_rota = Custo\_rota - \gamma(escolhido)$;
fim enquanto

Figura 5.6: Algoritmo *SoluçãoInicial_DROP*

economia associada à sua remoção. Enquanto for possível obter uma economia positiva, um novo vértice é escolhido para ser removido da rota corrente. Note que, como à rota contém inicialmente todos os vértices do grafo, acabam sendo removidos aqueles que acarretam os maiores acréscimos na distância total percorrida pela rota. O algoritmo associado o procedimento descrito nesta seção é mostrado na Figura 5.6.

A linha 5 inicializa a lista de candidatos $LDisp$ com todos os vértices de $Rota$ excluindo os vértices obrigatórios, já que estes devem fazer parte de toda solução viável do PRRG. A Figura 5.7 ilustra, passo-a-passo, o algoritmo *SoluçãoInicial_DROP* utilizando a instância PRRG representada pelo grafo da Figura 4.14.

Algoritmo SoluçãoInicial_DJK

Este algoritmo constrói uma solução inicial elemento a elemento, associando a cada iteração uma lista de candidatos.

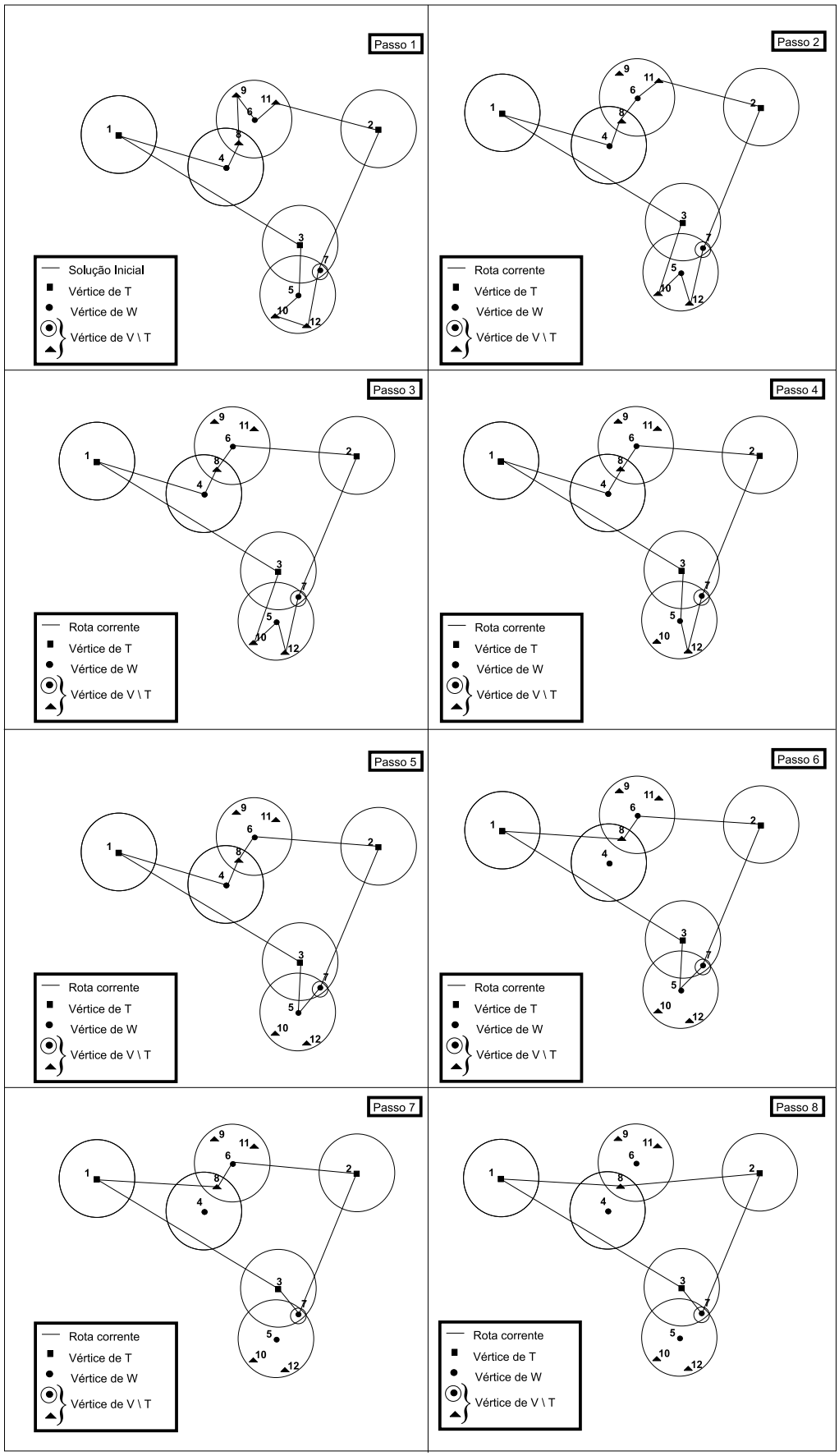

Figura 5.7: Passos de uma execução do algoritmo SoluçãoInicial_DROP

Considere o PRRG descrito como o Problema do Caixeiro Viajante Generalizado (seção 2.2), onde o conjunto de vértices do grafo é particionado em um conjunto de grupamentos definidos em (2.1) e (2.2). Seja $NSeq$ o conjunto de todos estes grupamentos. O algoritmo $SolucaoInicial\_DJK$ obtém uma solução inicial para o PRRG através de três fases: a primeira realiza o seqüenciamento do conjunto $NSeq$ e a segunda, encontra a melhor rota a partir dos grupamentos seqüenciados na fase anterior. A terceira fase é responsável pela eliminação de possíveis subciclos formados na segunda fase.

A primeira fase inicia-se com a escolha do primeiro grupamento da seqüência, que é selecionado aleatoriamente entre os grupamentos G_t, definidos por (2.2). A partir do primeiro grupamento seqüenciado, os demais são escolhidos iterativamente em uma lista de candidatos relacionada ao grupamento seqüenciado mais recentemente. Esta lista é construída de acordo com a ordem crescente da distância entre o grupamento seqüenciado mais recentemente e os demais grupamentos ainda não seqüenciados. A distância entre grupamentos é definida como:

$$Dist(G_i, G_j) = max\{c_{ab} \mid a \in G_i \, e \, b \in G_j\} \tag{5.3}$$

Se o número de grupamentos desta lista for muito elevado, uma lista de candidatos restrita (LCR) é construída. Quando não existirem mais grupamentos a serem seqüenciados, o algoritmo inicia a segunda fase.

A segunda fase tem a finalidade de definir que elemento (somente um) de cada grupamento pertencerá à rota. A idéia apresentada por Maniezzo et al. [13] é dispor todos os grupamentos em seqüência, duplicando o primeiro grupamento para simular uma lista circular. A partir desta permutação, um algoritmo de caminho mínimo fornece uma solução para o PRR.

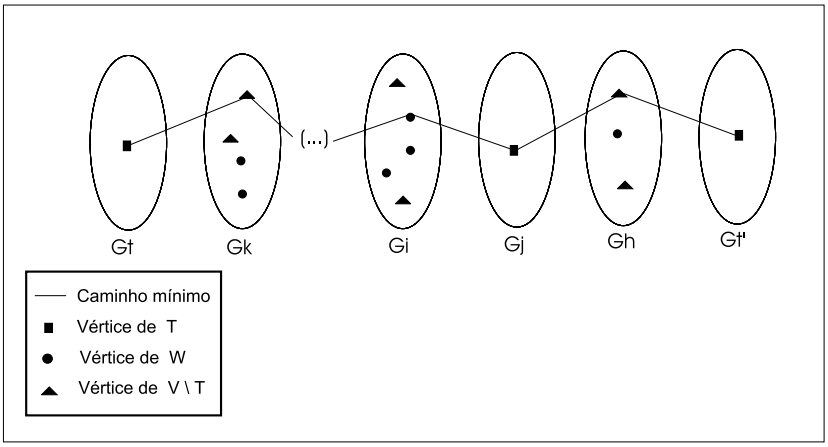

Figura 5.8: Seqüência dos Grupamentos de um grafo com uma solução viável para o PRRG.

Considere uma permutação de grupamentos obtida na primeira fase do algoritmo $(G_t, G_k, ..., G_i, G_j, G_h, G'_t)$ (representada na Figura 5.8), onde G'_t é uma cópia do primeiro grupamento ilustrado por G_t. De acordo as equações (2.1) e (2.2), existe um grupamento associado para cada $i \in W$ e para cada $j \in T$. Desta forma, como todos os grupamentos relacionados aos vértices de T, possuem apenas o vértice obrigatório que lhe dá origem e como cada vértice do conjunto W, origina um grupamento com todos os possíveis candidatos à sua cobertura, encontrar uma solução viável para o PRRG consiste em obter um caminho através dos grupamentos seqüenciados, tomando apenas um vértice em cada grupamento. Pode-se dizer ainda, que a melhor solução derivada de qualquer seqüência pode ser obtida computando o caminho mínimo a partir do vértice do primeiro grupamento até a sua respectiva cópia associada ao grupamento duplicado.

O algoritmo $SolucaoInicial\_DJK$ é uma adaptação da idéia apresentada em [13] e é descrito Figura 5.9.

A escolha inicial de um grupamento definido sob os vértices de T (linha 3), não afeta a busca pela melhor solução, já que uma lista circular é utilizada. Na linha 9, a Lista de Candidatos Restrita (LCR) é construída a partir de $NSeq$, ordenando os grupamentos de forma crescente de acordo com a distância entre estes e $clust\_atual$,

Algoritmo 6 : *SolucaoInicial_DJK*

1 **PRIMEIRA FASE:** Seqüenciamento dos Grupamentos
2 Inicializar $NSeq$ com todos os grupamentos do grafo;
3 Selecionar aleatoriamente um cluster $G_t \in NSeq$, com G_t definido sob os vértices de T;
4 $NSeq = NSeq - \{G_t\}$;
5 $clust\_atual = G_t$;
6 $Seq = \emptyset$;
7 $Seq = Seq \cup \{clust\_atual\}$;
8 **enquanto** $NSeq \neq \emptyset$ **faça**
9 Construir a Lista de Candidatos Restrita, LCR($clust\_atual$), sob os grupamentos de $NSeq$;
10 $clust\_selecionado$ = cluster selecionado em LCR($clust\_atual$);
11 $NSeq = NSeq - \{clust\_selecionado\}$;
12 $Seq = Seq \cup \{clust\_selecionado\}$;
13 $clust\_atual = cust\_selecionado$;
fim enquanto
14 **SEGUNDA FASE:** Construção da rota
15 Construir a rota computando o caminho mínimo através de uma adaptação do algoritmo de Dijkstra para caminho mínimo;
16 **TERCEIRA FASE:** Eliminação de subciclos formados na fase anterior

Figura 5.9: Algoritmo $SolucaoInicial\_DJK$.

determinada por 5.3.

Na segunda fase do algoritmo, o caminho mínimo é obtido através de uma adaptação do algoritmo de Dijkstra para caminhos mínimos [35]. Ao término desta fase, soluções inviáveis podem ser obtidas, pois não sendo disjuntos entre si, um grupamento pode conter um vértice também pertencente a outros grupamentos da seqüência, ocasionando a possível formação de subciclos. Desta forma, é necessário a existência de um procedimento que elimine estes subciclos, tornando a solução viável. Este processo é realizado na terceira fase e adota o critério de eliminar os ciclos que apresentam o maior acréscimo na distância total percorrida pela rota.

5.2.2 Algoritmos de Busca Local

A busca local é normalmente a fase mais dispendiosa de uma meta-heurística em termos do tempo computacional exigido. É nesta fase que vários ótimos locais são obtidos e analisados com o objetivo de tentar atingir um ótimo global. Esta seção apresenta quatro algoritmos de busca local para o PRRG. Dois destes algoritmos consideram o PRRG formulado como o Problema do Caixeiro Viajante Generalizado, descrito anteriormente na seção 2.2.

Algoritmo *Busca_ADD*

Baseado nos critérios de inserção e remoção já mencionados anteriormente na seção 5.2.1, o algoritmo *Busca_ADD* inicia a sua execução a partir de uma solução viável associada a uma seqüência de visitas.

Seja $LDisp$ a lista dos vértices não utilizados na solução inicial e seja $\{i, j\}$ um par de vértices desta lista ($i \neq j$). O algoritmo $Busca\_ADD$ insere o vértice i na rota inicial de modo a acarretar o menor acréscimo na distância total percorrida. Após a inserção do vértice i, o vértice j é inserido na nova rota usando o mesmo critério. Tendo a rota inicial acrescida dos vértices i e j, inicia-se um processo de remoções sucessivas, que ocorrem enquanto for possível obter uma redução no custo final associado a esta rota. Quando nenhuma melhora puder ser obtida, o custo da rota resultante é comparado ao custo da rota inicial. Caso este seja maior, a rota resultante é armazenada, senão ela é descartada. A rota inicial é retomada e um novo par de vértices de $LDisp$ é inserido. Ao final, o custo da nova rota é comparado ao custo da rota armazenada. Se o custo da nova rota for menor que o custo da rota armazenada, esta é substituída. O algoritmo prossegue até que todos os pares de vértices de $LDisp$ tenham sido inseridos. A Figura 5.10 descreve a $Busca\_ADD$. A Figura 5.11 ilustra, passo-a-passo, uma iteração do algoritmo $Busca\_ADD$. A solução inicial utilizada é aleatória e o par de vértices selecionado de $LDisp = \{4, 6, 7, 10, 12\}$ para inserção é dado por $(4, 10)$.

Algoritmo $Busca\_DROP$

O algoritmo de busca descrito nesta seção é similar ao descrito na seção anterior, pois baseia-se nos mesmos critérios de inserção e remoção de vértices. A diferença é queaqui a busca inicia-se removendo da solução inicial, o par de vértices (i, j), com $i \neq j$, que apresentar o menor prejuízo, já que a remoção ocorre mesmo que a ausência destes vértices inviabilize a solução.

A $Busca\_DROP$ se inicia com a remoção de um par de vértices $\{i, j\}$)

Algoritmo 7 : $Busca\_ADD$

1 $melhor\_rota = rota\_inicial$;
2 $melhor\_custo = custo\_inicial$;
3 **para** $(i, j) \in LDisp$ **faca**
4 $rota\_atual = rota\_inicial$;
5 $custo\_atual = custo\_inicial$;
6 Adicione i em $rota\_atual$ de acordo com a maior economia dada por ρ;
7 $custo\_atual = custo\_atual - \rho(i, posicao)$;
8 Adicione j em $rota\_atual$ de acordo com a maior economia dada por ρ;
9 $custo\_atual = custo\_atual - \rho(j, posicao)$;
10 **enquanto** for possível reduzir $custo\_atual$ removendo $k \in rota\_atual$ **faca**
11 Remova k de $rota\_atual$;
12 $custo\_atual = custo\_atual - \gamma(k)$;
fim enquanto
13 **se** $custo\_atual < melhor\_custo$
14 $melhor\_rota = rota\_atual$
15 $melhor\_custo = custo\_atual$
fim se
fim para

Figura 5.10: Algoritmo $Busca\_ADD$.

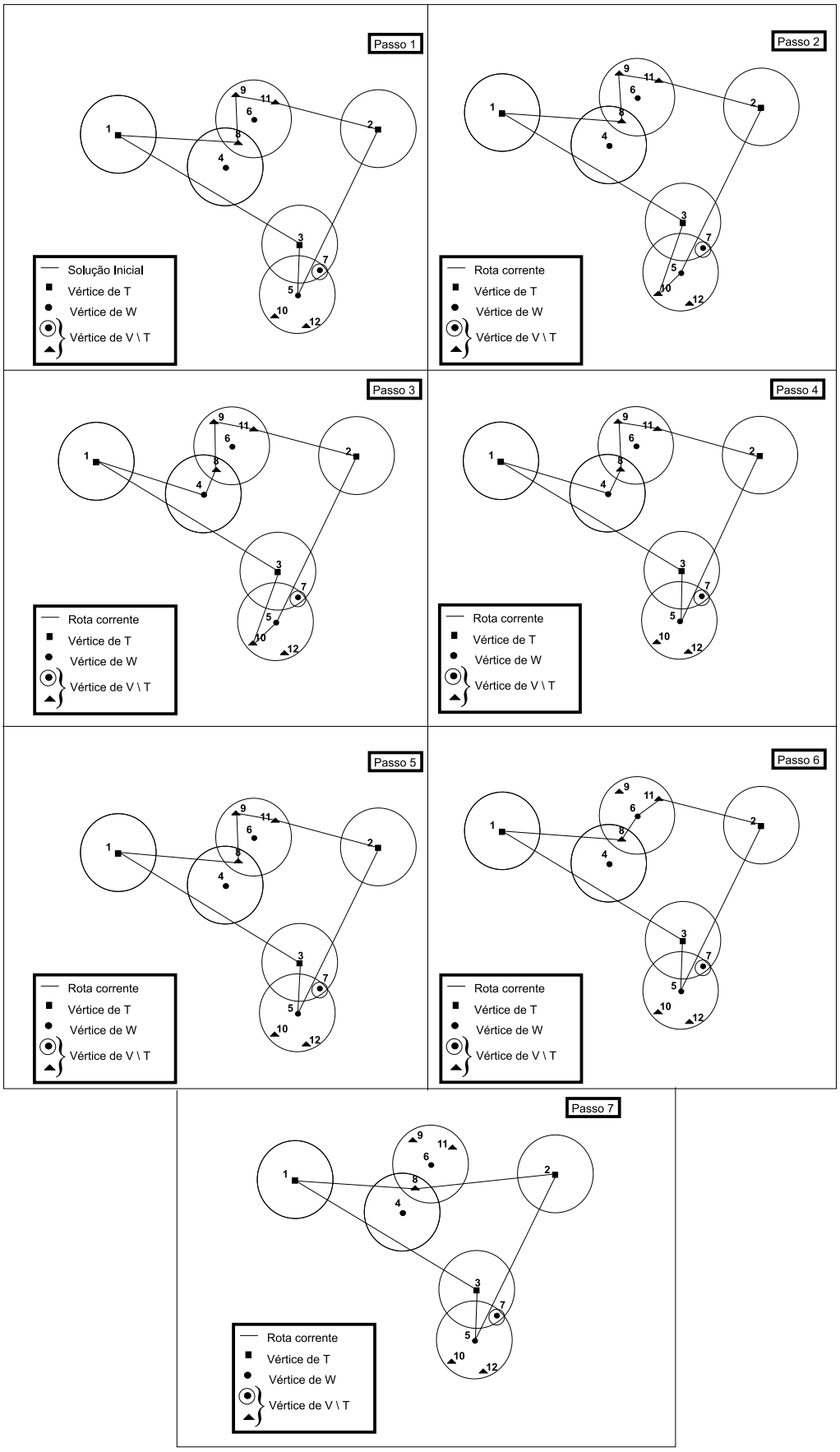

Figura 5.11: Passos da execução de uma iteração do algoritmo *Busca_ADD*.

da rota inicial, seguida de um processo de adições sucessivas que ocorre enquanto for possível encontrar uma melhora no custo final associado. A lista dos vértices candidatos para a adição ($LDisp$) é composta por todos os vértices n ao utilizados na rota inicial mais os removidos durante a execução do algoritmo. Estes vértices são considerados em $LDisp$ pois podem ser as melhores opções na redução do custo da rota, ou até mesmo responsáveis pela viabilidade da solução, como é o caso dos vértices obrigatórios. Quando nenhuma melhora puder ser obtida, o custo da rota resultante é comparado ao custo da rota inicial. Caso este seja maior, a rota resultante é armazenada, senão ela é descartada. A rota inicial é retomada e um novo par de vértices desta é removido, reiniciando o processo de adições. Ao final, o custo da nova rota é comparado ao custo da rota armazenada. Se o custo da nova rota for menor que o custo da rota armazenada, esta é substituída. O algoritmo prossegue até que todos os pares de vértices da rota inicial tenham sido removidos. O algoritmo $Busca\_DROP$ é apresentado na Figura 5.12.

A Figura 5.13 ilustra uma iteração do algoritmo $Busca\_DROP$. A solução inicial é a mesma utilizada para ilustrar o algoritmo $Busca\_ADD$. a e o par de vértices selecionado de $rota\_atual$ para a remoção é dado por $(5, 8)$.

Algoritmo $Busca\_VNS$

O algoritmo de busca local aqui proposto baseia-se nos conceitos da meta-heurística VNS, onde uma troca sistemática de estruturas de vizinhanças é realizada durante a busca, com o objetivo de se tentar escapar de ótimos locais distantes de um ótimo global. As vizinhanças da solução atual são exploradas incrementalmente quando uma melhora puder ser realizada. A grande vantagem desta abordagem, é a

Algoritmo 8 : *Busca_DROP*

1 $melhor\_rota = rota\_inicial$;
2 $melhor\_custo = custo\_inicial$;
3 **para** $(i, j) \in rota\_atual$ **faca**
4 $rota\_atual = rota\_inicial$;
5 $custo\_atual = custo\_inicial$;
6 Remova i de $rota\_atual$ de acordo com o menor prejuízo dado por γ;
7 $custo\_atual = custo\_atual - \gamma(i)$;
8 Remova j de $rota\_atual$ de acordo com o menor prejuízo dado por γ;
9 $custo\_atual = custo\_atual - \gamma(j)$;
10 $LDisp =$ todos os vértices não pertencentes a $rota\_atual$;
11 **enquanto** for possível reduzir $custo\_atual$ adicionando $k \in LDisp$ em $rota\_atual$ **faca**
12 Adicione k em $rota\_atual$;
13 $LDisp = LDisp - k$;
14 $custo\_atual = custo\_atual - \rho(k, posicao)$;
fim enquanto
15 **se** $custo\_atual < melhor\_custo$
16 $melhor\_rota = rota\_atual$
17 $melhor\_custo = custo\_atual$
fim se
fim para

Figura 5.12: Algoritmo *Busca_DROP*.

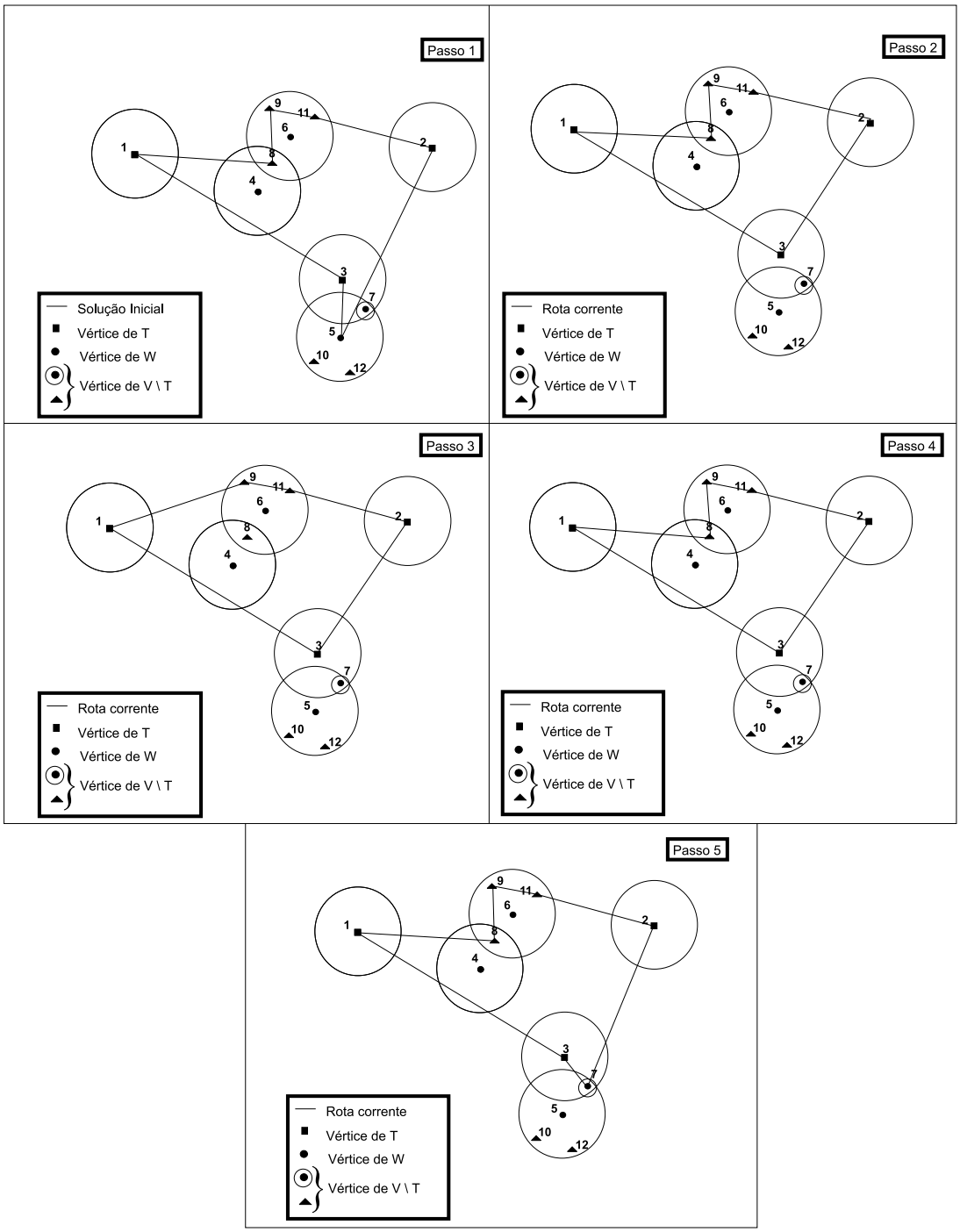

Figura 5.13: Passos da execução de uma iteração do algoritmo *Busca_DROP*.

utilização das características da solução corrente para a obtenção de soluções vizinhas promissoras.

O algoritmo $Busca\_VNS$ difere dos dois algoritmos de busca local apresentados anteriormente, por basear-se na estrutura de grupamentos associada ao grafo. A solução inicial desta busca é caracterizada por uma rota viável associada a uma seqüência de grupamentos, assim como a solução obtida pelo Algoritmo $SolucaoInicial\_DJK$.

Seja $S = G_t, G_k, ..., G_i, G_j, G_h, G'_t$, uma solução viável do PRRG e seja (G_i, G_j), com $i \neq j$, um par de grupamentos quaisquer desta solução, excluindo o grupamento origem G_t e o destino G'_t. Uma permutação em S é definida pela troca do grupamento G_i, com o grupamento G_j, na seqüência original associada à S. Desta forma, as vizinhanças de busca associadas a uma solução são definidas do seguinte modo:

- $V_1 = \{$Efetuar uma permutação (G_i, G_j), com $i \neq j$, na solução atual$\}$;
- $V_2 = \{$Efetuar duas permutações (G_i, G_j) e (G_l, G_k), com $i \neq k$ e $j \neq l$, na solução atual$\}$;
- $V_\alpha = \{$Efetuar α permutações (G_i, G_j), (G_l, G_k),...,(G_w, G_z) com os grupamentos diferentes entre si $\}$;

onde o valor de α pode variar até $(|T| + |W|)$, isto é, $\alpha \leq (n^o.\, de\, grupamentos) + 1)$.

A algoritmo $Busca\_VNS$ é apresentado na Figura 5.14.

Uma diferença entre o algoritmo $Busca\_VNS$ e o conceito original da

Algoritmo 9 : $Busca\_VNS$

```
1  Definir o conjunto de estruturas de vizinhanças V_α;
2  rota_inicial= solução inicial viável para o PRRG, com uma
   seqüência de grupamentos associada;
3  rota_atual = rota_inicial;
4  melhor_rota = rota_atual;
5  custo_atual = custo associado à rota_inicial;
6  melhor_custo = custo_atual;
7  enquanto critério de parada não satisfeito faça
8     α = 1
9     enquanto α < α_max faça
10       Gerar aleatoriamente uma solução rota_nova na vizinhan-
         ça V_α de rota_atual;
11       rota_atual = rota_nova;
12       custo_atual=custo associado a rota_nova;
13       se custo_atual < melhor_custo então
14          melhor_rota = rota_atual;
15          melhor_custo = custo_atual;
         senão
16          α = α + 1
         fim se
      fim enquanto
   fim enquanto
```

Figura 5.14: Algoritmo $Busca\_VNS$.

metaheurística VNS reside na forma de explorar as vizinhanças. A proposta de Hansen e Mladenović [32, 33] é explorar todas as soluções da vizinhança e selecionar a melhor. Ao contrário, no algoritmo proposto, as vizinhanças são exploradas parcialmente. O número de soluções investigadas em cada vizinhança é determinado por um valor proporcional ao número da vizinhança explorada, permitindo uma maior diversificação das soluções investigadas durante a execução do algoritmo e uma economia do tempo dispendido em relação a uma busca exaustiva dentro de uma estrutura de vizinhança.

Algoritmo $Busca\_OPT2$

Esta seção apresenta uma adaptação da heurística 2-Optimal [36], cuja idéia básica é realizar todas as combinações possíveis de trocas de duas arestas não adjacentes em uma determinada solução viável, tendo ao final do processo a melhor combinação encontrada.

Assim como no algoritmo $Busca\_VNS$, o algoritmo $Busca\_OPT2$ baseia-se na estrutura de grupamentos associada ao PRRG. A solução inicial desta busca é caracterizada por uma rota viável associada à uma seqüência de grupamentos, assim como a solução obtida pelo Algoritmo $SolucaoInicial\_DJK$. A partir desta seqüência, realiza-se todas as possíveis trocas nas posições de dois grupamentos. A cada permutação realizada, o custo associado é calculado pela adaptação do algoritmo de Dijkstra para caminho mínimo usado na segunda fase do algoritmo $SolucaoInicial\_DJK$ (seção 5.2.1). O algoritmo $Busca\_OPT2$ é apresentado na Figura 5.15.

O algoritmo $Busca\_OPT2$ termina quando todas as trocas de grupamentos dois a dois de $rota\_atual$ tiverem sido avaliadas. O primeiro e o último grupamento da seqüência não são considerados nas permutações. Ao final da busca, a melhor solução obtida entre todas as trocas realizadas é considerada como a nova solução.

Algoritmo 10 : $Busca\_OPT2$

1 $rota\_atual = rota\_inicial$;
2 $custo\_atual = custo\_inicial$;
3 $melhor\_rota = rota\_atual$;
4 $melhor\_custo = custo\_atual$;
5 **para** $(G_i, G_j) \in rota\_atual$ **faca**
6 Efetuar uma permutação (G_i, G_j), com $i \neq j$, na solução atual gerando $rota\_nova$;
7 $custo\_novo$=custo associado à $rota\_nova$;
8 **se** $custo\_novo < melhor\_custo$ **então**
9 $melhor\_rota = rota\_nova$;
10 $melhor\_custo = custo\_novo$;
fim se
fim para

Figura 5.15: Algoritmo $Busca\_OPT2$

5.2.3 Algoritmos GRASP para o PRRG

Baseada nas propostas dos algoritmos de construção e de busca local descritos respectivamente nas seções 5.2.1 e 5.2.2, esta seção apresenta oito (8) variações de metaheurísticas para solucionar aproximadamente o PRRG, ambas fundamentadas nos princípios básicos da metaheurística GRASP. A Tabela 5.1 apresenta um esquema com todas as oito variações.

Metaheurística GRASP	**Solução Inicial**	**Busca Local**
GRASP_DJ1	SolucaoInicial_DJK	Busca_VNS
GRASP_DJ2	SolucaoInicial_DJK	Busca_OPT2
GRASP_DJ3	SolucaoInicial_DJK	Busca_ADD
GRASP_DJ4	SolucaoInicial_DJK	Busca_DROP
GRASP_AA	SolucaoInicial_ADD	Busca_ADD
GRASP_AD	SolucaoInicial_ADD	Busca_DROP
GRASP_DA	SolucaoInicial_DROP	Busca_ADD
GRASP_DD	SolucaoInicial_DROP	Busca_DROP

Tabela 5.1: Metaheurísticas GRASP propostas para o PRRG

As quatro primeiras metaheurísticas apresentadas na Tabela 5.1, obtém a solução inicial através do algoritmo $SolucaoInicial\_DJK$, o único algoritmo de construção que opera na seqüência de grupamentos e não diretamente nos vértices do grafo. Analogamente, os algoritmos $Busca\_VNS$ e $Busca\_OPT2$, que compoem as metaheurísticas $GRASP\_DJ1$ e $GRASP\_DJ2$, são os únicos que também operam com uma seqüência de grupamemtos associados ao grafo.

Capítulo 6

Implementação e Resultados Computacionais

Embora seja um problema com inúmeras aplicações, o PRR é uma generalização do Problema do Caixeiro Viajante (PCV) ainda pouco explorado pela literatura afim. Além disso, de nosso conhecimento, não existe nenhuma biblioteca pública de problemas testes para o PRR. Logo, para avaliar o impacto das regras de redução e os algoritmos propostos neste trabalho, os problemas testes foram gerados da seguinte forma: os $|V| + |W|$ vértices foram tomados aleatoriamente em um retângulo de dimensão [560x400]; o conjunto T (vértices obrigatórios) foi definido considerando os $|T|$ primeiros pontos plotados no retângulo e W, os $|W|$ seguintes. A cardinalidade de cada um dos conjuntos T, W e V, foi escolhida aleatoriamente.

Para a análise das regras de redução, foram utilizadas duas estratégias. A primeira utiliza um método exato para verificar o impacto do conjunto das regras propostas na resolução de instâncias pequenas (grafos com até 25 vértices). Uma formulação matemática foi desenvolvida para o PRRG, introduzindo variáveis de fluxo z_{ij} para tratar as restrições que proíbem a formação de ciclos desconexos da origem. A formulação proposta foi implementada no software XPRESS (versão 11) [37], com es-

tratégia de corte ativada, disponível no laboratório de otimização da COPPE/Sistemas da UFRJ. A segunda estratégia utiliza as metaheurísticas propostas com grafos de até 300 vértices, comparando o desempenho de cada uma no grafo original e no grafo reduzido.

Para a análise das metaheurísticas propostas (veja a tabela 5.1), foram efetuados um número significativo de testes com a finalidade de comparar empiricamente que características inerentes a cada versão melhor se adequam ao PRRG. Os algoritmos foram implementados na linguagem C e os testes efetuados em um micro computador com processador AMD/K6II-450Mz e 128Mb de memória RAM.

Os testes computacionais estão classificados da seguinte forma:

1. Bateria de testes para pequenas instâncias (até 25 vértices) mostrando o impactos das Regras de Redução em um método exato;

2. Bateria de testes para instâncias com até 300 vértices, comparando o impacto das regras de redução nos algoritmos GRASP propostos, assim como uma análise comparativa do desempenho destas metaheurísticas.

6.1 Testes com o Método Exato

Esta seção apresenta os resultados obtidos com o método exato disponível no software XPRESS - versão 11 [37], com estratégia de corte ativada, utilizando instâncias para o PRRG com até 25 vértices. A formulação matemática proposta foi utilizada para implementar o problema em estudo.

A Tabela 6.1 apresenta os resultados relacionados ao tempo de execução do método exato (em segundos), utilizando tanto o grafo original como o grafo reduzido. Nesta, estão listados os tempos médios dispendidos na obtenção da solução ótima do PRRG para quatro grupos de nove instâncias. Os grupos foram divididos pelo número de vértices do grafo. Os resultados individuais, assim como as características do conjunto de vértices de cada instância, antes e após a redução, estão relacionados na seção 6.3.

$\|(V \bigcup W)\|$	Grafo Original	Grafo Reduzido
10	5,6	0,1
15	1239	7,8
20	**	203,4
25	**	4759,9

Tabela 6.1: Tempo requerido pelo método exato para obter a solução ótima em grafos de 10, 15, 20 e 25 vértices.

A primeira coluna mostra a cardinalidade do grupo de grafos testados. A segunda coluna apresenta o tempo médio dispendido na obtenção da solução ótima nos grafos originais e a terceira, o tempo médio para a obtenção da solução ótima nos grafos reduzidos. Os (**) indicam que o método exato não obteve a solução ótima após 4 horas de processamento. Pode-se observar, para os casos em que $|(V \bigcup W)|$ é igual à 20 e 25, só foi possível obter a solução ótima nos grafos reduzidos, o que justifica a utilização das regras de redução propostas neste método.

A Tabela 6.2 apresenta os tempos médios obtidos pelo método exato com os grafos reduzidos (segunda coluna) e por quatro metaheurísticas propostas (demais

Instâncias	Ex-RED	GRASP_Dj1	GRASP_Dj2	GRASP_Dj3	GRASP_Dj4
20_i	0	0	0	0	0
20_{ii}	0	0	0	0	0
20_{iii}	17	7,1	10,6	8,1	9,7
20_{iv}	13	8,9	11,8	7,6	8,9

Tabela 6.2: Tempo requerido pelo método exato para obter a solução ótima em grafos de 20 vértices.

colunas). Os tempos foram tomados em segundos e representam a média aritmética de cinco execuções. De modo geral, as metaheurísticas obtiveram um dempenho melhor que o método exato, encontrando a solução ótima em menos tempo.

6.2 Testes com as Metaheurísticas

Esta seção apresenta os resultados computacionais obtidos com os algoritmos GRASP propostos, considerando instâncias de 30 à 300 vértices.

A Tabela 6.3 mostra as características do conjunto de vértices, antes e após o uso das regras de redução propostas, para cada uma das instâncias utilizadas nos testes desta seção.

Grafo	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$ Reduzidos
G30a	30/7/19/4	21/7/6/8
G30b	30/10/10/10	19/10/3/6
G30c	30/12/12/6	21/12/6/3
G30d	30/19/7/4	22/19/2/1
G50a	50/10/30/10	34/10/13/11
G50b	50/30/10/10	30/30/0/ 0
G50c	50/20/20/10	30/20/8/2
G100a	100/20/40/40	57/20/16/21
G100b	100/40/20/40	48/40/2/6
G100c	100/35/35/30	51/35/7/9
G200a	200/60/100/40	77/60/5/12
G200b	200/80/80/40	87/80/1/6
G200c	200/100/60/40	100/100/0/0
G300a	300/90/140/70	90/90/0/0
G300b	300/100/100/100	117/100/4/13
G300c	300/140/90/70	140/140/0/0

Tabela 6.3: Características dos grafos utilizados nos testes com as metaheurísticas.

As Tabelas 6.4 e 6.5 relacionam respectivamente a performance dos algoritmos utilizando os grafos originais e reduzidos. Esta performance é calculada da

seguinte forma: {[(Solução Média − Melhor Solução Média)×100]÷ Melhor Solução Média}. O critério de parada adotado nestas Tabelas é o tempo de execução de 1200 segundos.

Instâncias_Orig	GRASP_Dj1	GRASP_Dj2	GRASP_Dj3	GRASP_Dj4
G30a	3,75%	0,00%	0,77%	10,00%
G30b	17,10%	15,03%	0,00%	7,28%
G30c	28,29%	25,33%	0,00%	18,75%
G30d	15,10%	15,01%	0,00%	6,60%
G50a	40,25%	39,19%	0,00%	30,67%
G50b	14,28%	10,61%	0,00%	7,80%
G50c	19,19%	17,04%	0,00%	4,86%
G100a	50,40%	49,72%	0,00%	42,38%
G100b	16,49%	17,67%	0,00%	4,97%
G100c	30,62%	26,12%	0,00%	26,59%

Tabela 6.4: Performance das metaheurísticas com grafos originais de até 100 vértices.

Pode-se notar que as metaheurísticas GRASP_Dj3 e GRASP_Dj4 obtiveram os melhores resultados em cerca de 85% dos casos. Uma característica comum a estas duas metaheurísticas está na natureza do procedimento de busca utilizado: *Busca_ADD* e *Busca_DROP*. Ambos realizam a busca local considerando cada vértice da solução e não grupamentos de vértices como nas demais. Com isso, o bom desempenho destas metaheurísticas se deve possivelmente a facilidade de diversificarem a

Instâncias_Red	GRASP_Dj1	GRASP_Dj2	GRASP_Dj3	GRASP_Dj4
G30a	3,33%	0,00%	8,08%	9,87%
G30b	5,91%	5,45%	4,54%	0,00%
G30c	0,56%	3,10%	0,00%	1,38%
G30d	10,35%	6,70%	3,15%	0,00%
G50a	3,30%	0,00%	0,62%	5,45%
G50b	3,31%	0,67%	2,25%	0,00%
G50c	38,84%	33,36%	27,18%	0,00%
G100a	11,13%	12,55%	0,00%	4,47%
G100b	6,80%	4,28%	0,00%	7,91%
G100c	0,23%	2,44%	0,00%	4,74%

Tabela 6.5: Performance das metaheurísticas com grafos reduzidos (instâncias originais com até 100 vértices).

sua busca. O procedimento utilizado na obtenção da solução inicial é o mesmo para os quatro algoritmos.

A Figura 6.1 apresenta um comparativo das performances dos algoritmos relacionados nas Tabelas 6.4 e 6.5.

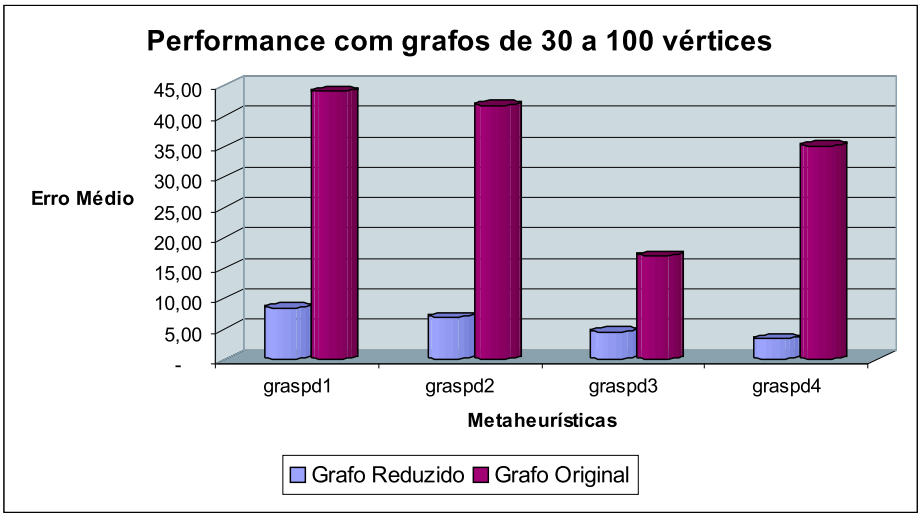

Figura 6.1: Impacto das regras de redução com grafos de 30 à 100 vértices.

Os resultados apresentados na Figura 6.1 constatam que o desempenho dos algoritmos no grafo original é consideravelmente pior que nos grafos reduzidos.

As Tabelas 6.6 e 6.7 mostram respectivamente a performance dos algoritmos GRASP_Dj1, GRASP_Dj2, GRASP_Dj3 e GRASP_Dj4 nos grafos originais e nos grafos reduzidos, com o critério de parada relacionado ao número de iterações realizadas por cada algoritmo (número de iterações $= (2 \times |W \cup W|)$). Observando cada uma destas Tabelas, constata-se que os algoritmos GRASP_Dj3 e GRASP_Dj4 obtiveram os melhores resultados. Este fato pode ser justificado pelo critério de parada adotado, uma vez que realizando iterações mais elaboradas, os algoritmos conseguem soluções de melhor qualidade, ao contrário do algoritmos GRASP_Dj1 e GRASP_Dj2, que obtiveram os piores resultados. Quando o critério de parada é o tempo de execução, como nas Tabelas 6.4 e 6.5, os algoritmos GRASP_Dj1 e GRASP_Dj1 realizam um número

Instâncias_Orig	GRASP_Dj1	GRASP_Dj2	GRASP_Dj3	GRASP_Dj4
G50a	46,68%	41,96%	0,00%	23,97%
G50b	33,50%	28,66%	4,92%	0,00%
G50c	23,53%	16,79%	0,00%	4,29%
G100a	70,51%	62,17%	0,00%	36,80%
G100b	41,25%	33,08%	9,83%	0,00%
G100c	50,29%	42,30%	0,00%	18,18%

Tabela 6.6: Performance das metaheurísticas com grafos originais de 50 e 100 vértices.

Instâncias_Red	GRASP_Dj1	GRASP_Dj2	GRASP_Dj3	GRASP_Dj4
G50a	34,48%	19,70%	0,00%	15,19%
G50b	14,12%	2,88%	14,56%	0,00%
G50c	64,05%	52,01%	53,48%	0,00%
G100a	19,37%	13,71%	0,00%	18,56%
G100b	17,32%	10,59%	21,20%	0,00%
G100c	16,65%	11,79%	0,71%	0,00%

Tabela 6.7: Performance das metaheurísticas com grafos reduzidos (instâncias originais de 50 e 100 vértices).

de iterações mais significativo, conseguindo melhorar seus resultados. Pode-se notar também (Tabela 6.6) que o desempenho das metaheurísticas aumentam consideravelmente nos grafos reduzidos.

A Figura 6.2 apresenta um comparativo das performances dos algoritmos relacionados nas Tabelas 6.6 e 6.7. Novamente pode-se notar que o desempenho das metaheurísticas é muito melhor nos grafos reduzidos.

As Tabelas 6.8 e 6.9 mostram respectivamente a performance dos algoritmos utilizando os grafos originais (200 e 300 vértices) e reduzidos. O critério de parada adotado é o tempo de execução de 1200 segundos.

Apesar de utilizar o tempo de execução como critério de parada, os dois algoritmos que possuem a fase de busca local mais dispendiosa obtiveram os melhores resultados, mesmo na maioria das vezes realizando apenas uma iteração. Este fato pode ser justificado porque estes procedimentos eliminam mais freqüentemente

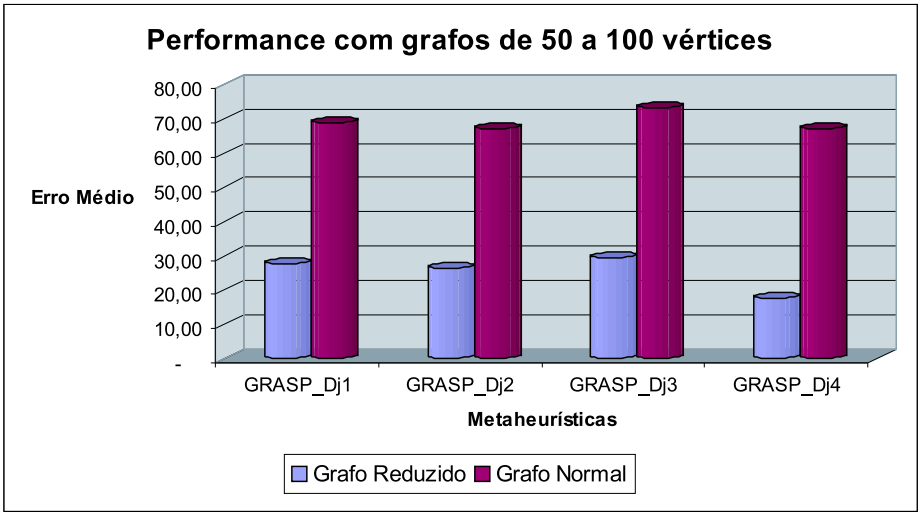

Figura 6.2: Impacto das regras de redução com grafos de 50 à 100 vértices.

Instâncias_Orig	GRASP_Dj1	GRASP_Dj2	GRASP_Dj3	GRASP_Dj4
G200a	29,91%	36,03%	0,00%	30,98%
G200b	19,52%	23,48%	0,00%	27,18%
G200c	7,31%	10,09%	0,00%	12,04%
G300a	26,10%	36,89%	0,00%	37,77%
G300b	28,56%	32,17%	0,00%	45,26%
G300c	16,33%	25,85%	0,00%	29,11%

Tabela 6.8: Performance das metaheurísticas com grafos originais de 200 e 300 vértices.

vértices desnecessários à solução viável do problema, ao contrário dos procedimentos de busca local que realizam a busca com permutações de grupamentos, onde os vértices desnecessários só são eliminados quando dois grupamentos não disjuntos são dispostos de forma adjacente.

A Figura 6.3 apresenta um comparativo das performances dos algoritmos relacionados nas Tabelas 6.8 e 6.9.

A Tabela 6.10 mostra a performance de cinco metaheurísticas propostas que utilizam procedimentos distintos na a construção de uma solução inicial para o PRRG. Os resultados obtidos confirmam o impacto da qualidade da solução inicial na solução final. Note que, apesar dos bons resultados obtidos nos demais testes, os algoritmos GRASP_Dj3 e GRASP_Dj4 podem melhorar muito o seu desempenho

Instâncias_Red	GRASP_Dj1	GRASP_Dj2	GRASP_Dj3	GRASP_Dj4
G200a	8,55%	7,80%	9,75%	0,00%
G200b	1,63%	4,08%	3,05%	0,00%
G200c	21,70%	25,39%	20,43%	0,00%
G300a	0,68%	4,35%	1,60%	0,00%
G300b	15,34%	17,57%	21,08%	0,00%
G300c	0,00%	4,12%	0,35%	1,86%

Tabela 6.9: Performance das metaheurísticas com grafos reduzidos (instâncias originais de 200 e 300 vértices).

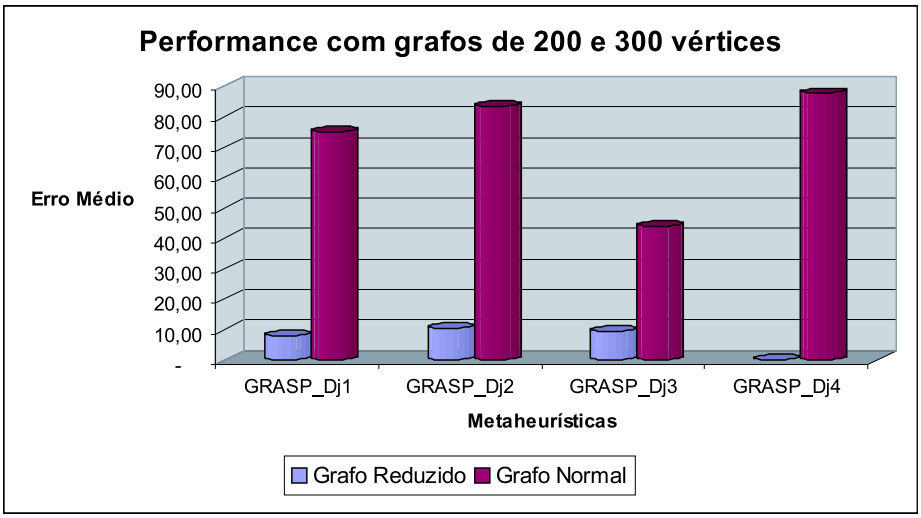

Figura 6.3: Impacto das regras de redução com grafos de 200 e 300 vértices.

quando conjugados a procedimentos para construção de solução inicial mais eficiente.

Instâncias_Red	GRASP_AA	GRASP_AD	GRASP_DD	GRASP_Dj3	GRASP_Dj4
G200a	1,23%	6,42%	0,00%	248,51%	356,47%
G200b	0,00%	2,98%	0,85%	251,44%	346,95%
G200c	0,00%	1,92%	0,02%	310,32%	359,71%

Tabela 6.10: Performance das metaheurísticas com procedimentos distintos para a construção de uma solução inicial.

6.3 Resultados adicionais do Método Exato

Este capítulo apresenta informaç oes adicionais aos resultados obtidos com o método exato apresentados no capítulo 6.

6.3.1 Características dos conjuntos de vértices

As Tabelas seguintes (6.11, 6.12, 6.13, 6.14) relacionan as características do conjunto de vértices de todos grafos utilizados nos testes com o método exato, antes e após a utilização das regras de redução propostas neste trabalho. Os (**) indicam que o método exato não obteve a solução ótima após 4 horas de processamento.

Grafo	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$ Reduzidos
G10a1	10/3/5/2	7/3/4/0
G10a2	10/3/5/2	6/3/3/0
G10a3	10/3/5/2	9/3/5/1
G10b1	10/4/4/2	8/4/3/1
G10b2	10/4/4/2	8/4/3/1
G10b3	10/4/4/2	7/4/3/0
G10c1	10/5/3/2	7/5/2/0
G10c2	10/5/3/2	8/5/3/0
G10c3	10/5/3/2	7/5/2/0

Tabela 6.11: Características dos grafos com 10 vértices

Grafo	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$ Reduzidos
G15a1	15/3/8/4	11/3/8/0
G15a2	15/3/8/4	11/3/7/1
G15a3	15/3/8/4	8/3/5/0
G15b1	15/6/6/3	12/6/5/1
G15b2	15/6/6/3	9/6/2/1
G15b3	15/6/6/3	12/6/5/1
G15c1	15/8/3/4	9/8/1/0
G15c2	15/8/3/4	13/8/3/2
G15c3	15/8/3/4	10/8/2/0

Tabela 6.12: Características dos grafos com 15 vértices

Grafo	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$ Reduzidos
G20a1	20/5/10/5	11/5/6/0
G20a2	20/5/10/5	15/5/8/2
G20a3	20/5/10/5	16/5/7/4
G20b1	20/8/8/4	11/8/3/0
G20b2	20/8/8/4	12/8/2/2
G20b3	20/8/8/4	14/8/4/2
G20c1	20/10/5/5	14/10/4/0
G20c2	20/10/5/5	12/10/2/0
G20c3	20/10/5/5	17/10/3/4

Tabela 6.13: Características dos grafos com 20 vértices

6.3.2 Tempos de execução do método exato

O tempo requerido (em segundos) pelo software XPRESS na obtenção da solução ótima, para cada uma das instâncias testadas, estão dispostos nas tabelas desta seção (6.16, 6.17, 6.18, 6.19).

Grafo	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$	$\|V \cup W\|$ / $\|T\|$ / $\|W\|$ / $\|V \backslash T\|$ Reduzidos
G25a1	25/7/10/8	18/7/7/4
G25a2	25/7/10/8	17/7/5/5
G25a3	25/7/10/8	14/7/5/0
G25b1	25/10/7/8	14/10/3/1
G25b2	25/10/7/8	20/10/5/5
G25b3	25/10/7/8	17/10/4/3
G25c1	25/10/10/5	18/10/5/3
G25c2	25/10/10/5	14/10/2/2
G25c3	25/10/10/5	22/10/8/4

Tabela 6.14: Características dos grafos com 25 vértices

Grafo	$\|V \cup W\|$	$\|T\|$	$\|W\|$	$\|V \cup W\|$ Red.	$\|T\|$ Red.	$\|W\|$ Red.
20_*i*	20	3	3	7	3	3
20_*ii*	20	4	4	9	4	4
20_*iii*	20	5	10	14	5	9
20_*iv*	20	10	5	12	10	2

Tabela 6.15: Características do segundo grupo de grafos com 20 vértices

Grafo	Sem Redução	Com Redução
G10a1	2	0
G10a2	16	0
G10a3	2	1
G10b1	3	0
G10b2	3	0
G10b3	0	0
G10c1	1	0
G10c2	13	0
G10c3	10	0

Tabela 6.16: Tempo do XPRESS com grafos de 10 vértices.

Grafo	Sem Redução	Com Redução
G15a1	182	9
G15a2	897	15
G15a3	159	0
G15b1	42	13
G15b2	1843	0
G15b3	289	13
G15c1	4831	1
G15c2	37	19
G15c3	2872	0

Tabela 6.17: Tempo do XPRESS com grafos de 15 vértices.

Grafo	Sem Redução	Com Redução
G20a1	824	0
G20a2	**	361
G20a3	**	272
G20b1	**	1
G20b2	**	2
G20b3	**	5
G20c1	**	2
G20c2	**	1
G20c3	**	1187

Tabela 6.18: Tempo do XPRESS com grafos de 20 vértices.

Grafo	Sem Redução	Com Redução
G25a1	**	**
G25a2	**	4528
G25a3	**	2
G25b1	**	171
G25b2	**	1262
G25b3	**	250
G25c1	**	7812
G25c2	**	14
G25c3	**	**

Tabela 6.19: Tempo do XPRESS com grafos de 25 vértices.

Capítulo 7

Conclusões e Trabalhos Futuros

Apresenta-se neste trabalho algumas contribuições para o Problema de Recobrimento de Rotas Generalizado.

Inicialmente é proposta uma formulação matemática descrevendo o PRRG como um Problema de Programação Linear Inteira, onde o problema de evitar a formação de ciclos desconexos do vértice origem é tratado introduzindo variáveis de fluxo. Como resultado, desenvolveu-se uma formulação que exibe um número menor de restrições que o utilizado nos métodos existentes para o PRR.

Outra contribuição é a proposta de adaptar as regras de redução existentes para o PRR considerando o modelo generalizado (PRRG). Nesta etapa, verifica-se que algumas regras existentes se tornam inviáveis. A partir das características do PRRG, uma nova regra é proposta. O impacto do conjunto de regras de redução para o PRRG é claro e definitivo, justificando plenamente a sua adoção.

Em termos de algoritmos heurísticos, existe um sentimento entre os pesquisadores da área de otimização de que os melhores são: a Busca Tabu (BT), o GRASP e o VNS. Como a BT já é uma técnica muito explorada pela literatura,

ao contrário das outras duas, neste trabalho optou-se por desenvolver novas metaheurísticas baseando-se nos conceitos do algoritmo GRASP. São sugeridas diferentes versões de um algoritmo GRASP para o PRRG. A maioria obteve resultados médios promissores, a menos da versão que utiliza conceitos de clusterização para os vértices de $T \cup W$ associados à algoritmos de caminho mínimo. Nestes casos, os resultados médios obtidos para instâncias maiores (veja Tabela 6.10) foram muito "pobres" quando comparados com o desempenho das outras metaheurísticas aqui propostas. Em parte, o desempenho destes algoritmos podem ser justificados pelo fato dos métodos utilizarem sempre um número de vértices igual ou bastante próximo de $|T \cup W|$ na solução inicial.

Finalizando, pode-se concluir que algoritmos já bem conhecidos, como os que usam conceitos de "ADD" e "DROP", se forem bem explorados em uma metaheurística GRASP, podem superar outras técnicas, assim como mostram os resultados da Tabela 6.10. Além disso, as metaheurísticas do tipo GRASP podem ser usadas com sucesso na solução do PRR ou PRRG e em problemas similares, mesmo usando técicas simples de construção e busca local, como as propostas neste trabalho.

Como trabalhos futuros podemos incluir:

- Investigar novas regras de redução, tanto para o PRRG como para problemas similares. Uma das propostas é utilizar o conceito de envoltória convexa para eliminar determinados vértices em análises posteriores;
- Avaliar o desempenho de algoritmos GRASP que utilizam o conceito de clusterização dos vértices de $(T \cup W)$, incorporando outras técnicas para a remoção de vértices ainda na solução inicial;
- Adaptar as contribuições aqui propostas para os demais problemas de otimização combinatória, onde a solução utiliza apenas um subconjunto de vértices do grafo

de entrada;

- Paralelizar os algoritmos aqui propostos.

Artigos Publicados

Título: Uma metaheurística GRASP/VNS para uma solução aproximada do Problema de Recobrimento de Rotas
Autores: Luciene C. S. Motta, Luiz S. Ochi e Carlos A. Martinhon;
Local: *status*: aceito para publicação integral na série "TEMAS - Tendências em Matemática Aplicada e Computacional", vol.2, número 1, 2001;
Editora: SBMAC, ISBN: 85-86883-02-6;
Editores: E. X. L. Andrade, G. N. Silva e A. Sri Ranga.

Título: Reduction Rules for the Covering Tour Problem
Autores: Luciene C. S. Motta, Luiz S. Ochi e Carlos A. Martinhon;
Local: Eletronic Notes in Discrete Mathematics;
Editora: ISNN: 0166-218X;
Editores: J. L. Swarcfter and S. Song.

Título: GRASP Metaheuristic to the Generalized Covering Tour Problem
Autores: Luciene C. S. Motta, Luiz S. Ochi e Carlos A. Martinhon;
Local: *status*: aceito para apresentação e publicação nos Proc. of the IV Metaheuristic International Conference (MIC'2001), Porto-Portugal;
Editores: Ana Vianna andJ. P. Souza.

Título: O Problema de Recobrimento de Rotas: Formulação Matemática, Testes de Redução e Soluções Aproximadas via GRASP
Autores: Luciene C. S. Motta, Luiz S. Ochi e Carlos A. Martinhon;
Local: Anais do XXXII Simpósio Brasileiro de Pesquisa Operacional (XXXII SBPO), Viçosa-MG;
Editora: SOBRAPO, ISBN: 1518-1731, em CD-ROM pp:137-150, 2000.

Título: Metaheurística Híbrida GRASP + VNS para uma solução aproximada do Problema de Recobrimento de Rotas
Autores: Luciene C. S. Motta, Luiz S. Ochi e Carlos A. Martinhon;
Local: Anais do XXIII Congresso Nacional de Matemática Aplicada e Computacional (XXIII CNMAC), Santos-SP;
Editora: SBMAC, pp:296 (resumo), 2000.

Bibliografia

[1] J. H. Holand, *Adaptation in Natural an Artificial Systems*, University of Michigan Press, Ann Arbor, (1975).

[2] E. D. Taillard, *Ant Systems*, Technical Report IDSIA-05-99, (1999).

[3] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, (1990).

[4] J. Current, *Multiobjective Design of Transportation Networks*, Ph.D. Thesis, Department of Geography and Environmental Engineering, The Johns Hopkins University, (1981).

[5] J. Current and D. A. Schilling, *The Covering Salesman Problem*, Transportation Science 23 (1989), 208-213.

[6] J. Current, E. Rolland, *Efficient Algorithms for Solving the Shortest Covering Path Problem*, Transportation Science 28 (1994), 317-327.

[7] M. Gendreau, G. Laporte, and F. Semet, *The Covering Tour Problem*, Operations Research 45 (1995), 568-576.

[8] M. Gendreau, A. Hertz and G. Laporte, *New Insertion and Postoptimization Procedures for the Traveling Salesman Problem*, Opns. Res. 40 (1992), 1086-1094.

[9] E. Balas and A. Ho, *Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradiente Optimization: A Computacional Study*, Math. Programming, 12 (1980), 37-70.

[10] E. Balas, *The Prize Collecting Traveling Salesman Problem*, ORSA/TIMS (1986).

[11] G. Laporte and S. Martello, *The Selective Traveling Salesman Problem*, Discrete Appl. Math. 26 (1990), 193-207.

[12] M. Fischetti, J. J. Salazar and P. Toth, *A Branch-and-Cut Algorithm for the Symetric Generalized Traveling Salesman Problem*, Working Paper, DEIS, University of Bologna, (1994).

[13] V. Maniezzo, R. Baldacci, M. Boschetti and M. Zamboni, *Scatter Search Methods for The Covering Tour Problem*, Scienze dell'Informazione, University of Bologna, (1999).

[14] F. Glover, *Scatter Search and Star Paths: Beyond the Genetic Metaphor*, OR. Spektrum 17 (1995), 125-137.

[15] F. Glover, *A Template for Scatter Search and Path Relinking*, Lecture Notes in Computer Science, J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), (1997).

[16] CPLEX Optimizaton Inc., *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, (1993).

[17] J. Current, C. Revelle and J. Cohon, *The Shortest Covering Path Problem: An Application of Locational Constraints to Network Design*, Journal of Regional Sciences 24 (1984), 161-183.

[18] M. Hachicha, M. J. Hodgson and G. Laporte, *Heuristics for the multi-vehicle covering tour problem*, Computers and Operations Research 27 (2000), 29-42.

[19] J. Current and D. A. Schilling, *The Median Tour and Maximal Covering Tour Problem: Formulations and Heuristics*, European Journal of Operational Research 73(1994), 114-126.

[20] M. Labbe and G. Laporte, *Maximizing user convenience and postal service efficiency in post box location*, Belgian J. Opns. Res. Statist. and Computer Sci. 26 (1986), 21-35.

[21] E. Balas, *The Prize Collecting Traveling Salesman Problem*, Networks 19 (1989), 621-636.

[22] M. Fischetti and P. Toth, *An additive approach for the optimal solution of the prize collecting Traveling Salesman Problem*, in Vehicle Routing: Methods and Studies, B.L. Golden and A.A. Assad (eds.), North-Holland, Amsterdam (1988), 319-343.

[23] J. R. Oppong and M. J. Hodgson, *Spatial acessibility to health care facilities in Suhum District, Ghana*, Professional Geographer 46 (1994), 199-209.

[24] E. M. Arkin and R. Hassin, *Approximating Algorithms for the Geometric Covering Salesman Problem*, Discrete Appl. Math. 55 (1994), 197-218.

[25] S. Ntafos, *Watchman Routes under Limited Visibility.*, Computational Geometry 1, 149-170.

[26] P. Kourtz, *The need for improved forest fire detection*, Forestry Chronicle, 272-277, (1987).

[27] C. S. Revelle and G. Laporte, *New directions in plant locations*, Studies in Locational Analysis 5 (1993), 31-58.

[28] G. Finke, A. Claus, and E. Gunn, *A two-commodity network flow approach to the Traveling Salesman Problem*, Congress. Numerantium 41, 167-178, (1984).

[29] E. Aarts and J. Korst, *Simulated Anneling an Boltzmann Machines* , John Wiley & Sons, (1990).

[30] W. S. McCulloch and W. Pitts, *A logical calulus os the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics 5 (1943), 115-133.

[31] F. Glover, *Future Paths for Integer Programing and Links to Artificial Intelligence Problem*, Management Science 40/10 (1986), 1276-1290.

[32] P. Hansen and N. Mladenovìc, *Variable Neighborhood Search for the p-median*, Location Science 5 (1997), 207-226.

[33] P. Hansen, N. Mladenovìc and D. Perez-Brito, *Variable Neighborhood Decomposition Search*, GERAD (1998).

[34] M. G. C. Resende, and T. A. Feo, *Greedy Randomized Adaptative Search Procedures*, Journal of Global Optimization (1995), 1-27.

[35] Dijkstra, E., *A Note on two Problems in Connexion with Graphs*, Numeriche Mathematics 1, (1959), 269 - 271.

[36] S. Lins and B. W. Kernighan, *An Effective Heuristic Algorithm for the Traveling Salesman Problem*, Oper. Res. 21, (1973), 498-516.

[37] Dash, Biswoth and Northants, *XPRESS-MP*, NN7 3BX, UK, (1994).