O Problema de Recobrimento de Rotas com Coleta de Prêmios: Regras de Redução, Formulação Matemática e Heurísticas

Adria Ramos de Lyra

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização e Inteligência Artificial.

Orientador: Luiz Satoru Ochi

Niterói, Setembro de 2004.

O Problema de Recobrimento de Rotas com Coleta de Prêmios: Regras de Redução, Formulação Matemática e Heurísticas

Adria Ramos de Lyra

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização e Inteligência Artificial.

Aprovada por:

Prof. Luiz Satoru Ochi / IC-UFF (Presidente)

Prof. Carlile Campos Lavor / UERJ

Prof. Celso da Cruz Carneiro Ribeiro / IC-UFF

Prof. Fabio Protti / UFRJ

Profa. Lúcia Maria de Assumpção Drummond / IC-UFF

Niterói, Setembro de 2004.

A meus pais, irmão e avós,

com amor.

Agradecimentos

Agradeço primeiramente a Deus, por mais esta vitória. Porque Tua é a honra e a glória.

A meus pais, Maura e Carlos, meu irmão Junior, e avós, para os quais todo agradecimento sempre será pouco, por todo amor, carinho, apoio e encorajamento.

Ao meu orientador (desde a iniciação científica), Luiz Satoru Ochi, pela amizade, dedicação, pelo crescimento pessoal e profissional durante estes anos de convívio.

Ao Professor Nelson Maculan, da Universidade Federal do Rio de Janeiro, por ter gentilmente cedido o Laboratório de Otimização da COPPE-Sistemas (LabOtim) onde os testes com o XPRESS-MP foram realizados.

Aos professores Adilson Xavier, Jayme Szwarcfiter, Marcia Fampa e Sulamita Klein, da Universidade Federal do Rio de Janeiro, que permitiram que eu cursasse suas disciplinas no Programa de Pós-Graduação da COPPE-Sistemas (PESC).

Aos professores que aceitaram fazer parte da banca examinadora deste trabalho.

Aos amigos que fiz na COPPE, Carol, Elivelton, Fátima, Luidi, Rosa, Talita e Yuri que me proporcionaram um ambiente tão agradável de trabalho.

Aos amigos que fiz durante este mestrado na UFF que tornaram, sem dúvida, estes anos muito mais agradáveis.

Ao meu grande amigo Yuri Abitbol, pela ajuda na implementação, pela sua compania e incentivo na reta final da defesa.

A minha queridíssima amiga Fátima, pela revisão desta tese, pela amizade, pelas conversas e por todo o carinho com que me acolheu no LabOtim.

Aos amigos de todas as horas, Carlos Marcelo, Gabriel Junior , Leo, Miguel, Mônica e Xandinha, pela amizade incondicional e pela constante torcida.

À todos os funcionários do Instituto de Computação, em especial à Angela, à Isabela e ao Carlinhos.

Ao Instituto de Computação desta Universidade pela oportunidade de desenvolver meus estudos, agora no doutorado.

À Coordenação de Aperfeiçamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro.

Resumo da Tese apresentada à UFF como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação (M.Sc.)

O Problema de Recobrimento de Rotas com Coleta de Prêmios: Regras de Redução, Formulação Matemática e Heurísticas

Adria Ramos de Lyra

Setembro/2004

Orientador: Luiz Satoru Ochi
Programa de Pós-Graduação em Computação

Este trabalho propõe técnicas para resolver de forma eficiente uma generalização do Problema de Recobrimento de Rotas (PRR). Dado um grafo não direcionado e ponderado $G(V \cup W, A)$, o conjunto de vértices associado $V \cup W$ é composto pelos seguintes: $T \subseteq V$ (vértices obrigatórios), $V \backslash T$ (vértices optativos) e W, o conjunto de vértices a serem cobertos pela solução. O PRR consiste em encontrar um ciclo de custo mínimo que contenha todos os vértices de T, cubra todos os vértices de W utilizando, se necessário, vértices de $V \backslash T$. Este problema pode ser visto como uma extensão do Problema do Caixeiro Viajante (PCV); portanto, pertence à classe NP-Difícil. Neste trabalho abordamos uma generalização do PRR, que denotamos por: O Problema de Recobrimento de Rotas com Coleta de Prêmios (PRR-CP), onde cada vértice do conjunto V possui associado um valor positivo (prêmio), visando a obter uma rota do PRR que colete pelo menos um valor mínimo de prêmio. O objetivo deste trabalho é propor procedimentos para resolver de forma eficiente o PRR-CP. Para isso são propostos: (i) uma formulação matemática descrevendo o PRR-CP como um problema de programação linear inteira; (ii) um conjunto de regras de redução para os dados de entrada; e (iii) heurísticas utilizando conceitos de GRASP para obter bons limites superiores do valor ótimo.

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

The Prize Collecting Covering Tour Problem: Reduction Rules, Mathmatical Formulation and Metaheuristics

Adria Ramos de Lyra

September/2004

Advisors: Luiz Satoru Ochi
Department: Computer Science

This work proposes techniques to solve in an efficient way a generalization of the Covering Tour Problem (CTP). Starting with a graph $G(V \cup W, A)$, the set of associated vertexes $V \cup W$ is composed by the following subsets: $T \subseteq V$ (obligatory vertices), $V \backslash T$ (optional vertices) and W, the set of vertices that must be covered by the solution. CTP consists of finding a cycle of minimum cost that contains all the vertexes of T, cover all the vertexes of W, using, if necessary, vertexes of $V \backslash T$. This problem can be seen as an extension of the Traveling Salesman Problem (TSP), therefore, it belongs to the NP-hard. In this work we approached a generalization of CTP, that we denoted as: The Prize Collection Covering Tour Problem (PC-CTP). Where each vertex of the group V is associated to a positive value (prize) and the objective of PC-CTP is to obtain a route of PRR, collecting at least a minimum value of prize. The objective of this work, is to propose procedures to solve PC-CTP in a efficient way, for that we propose: (i) A mathematical formulation describing the PC-CTP as a problem of integer linear programming; (ii) a set of reduction rules for the entrance data; and (iii) heuristics using concepts of GRASP to obtain good superior limits of the best value.

Palavras-chave

1. Otimização Combinatória
2. Metaherísticas
3. Regras de Redução
4. O Problema de Recobrimento de Rotas

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O Problema de Recobrimento de Rotas (PRR) é um problema de otimização combinatória de elevada complexidade computacional, ainda pouco explorado pela literatura afim.

O objetivo é encontrar, em um grafo não direcionado e ponderado $G = (V \cup W, E)$, o ciclo de menor custo, obedecendo a um conjunto de restrições, onde $V \cup W = \{1, ..., m\}$ representa o conjunto de vértices, com $V \cap W = \emptyset$; $E = \{(i, j) \mid i, j \in V \cup W,\ i \neq j\}$, o conjunto de arestas; V, o conjunto de vértices que podem ser visitados. Além disso, $T \subseteq V$ é o conjunto dos vértices que devem ser visitados; W, o conjunto dos vértices que devem ser cobertos. 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 dado de entrada para o problema. 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.

O PRR é considerado NP-difícil uma vez que ele pode ser reduzido ao Problema do Caixeiro Viajante (PCV) quando $d = 0, \forall j \in V$ e $V = W$. [35]

Este trabalho apresenta um estudo do PRR, onde se propõe uma versão generalizada deste, denominada Problema de Recobrimento de Rotas com Coleta de Prêmios (PRR-CP), 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 PRR-CP. Propõem-se ainda heurísticas baseadas em conceitos do método GRASP (*Greedy Randomized Adaptive Search Procedure*) para a solução aproximada do problema.

Este trabalho apresenta-se da seguinte forma: no Capítulo 2, apresenta-se a descrição do Problema de Recobrimento de Rotas com Coleta de Prêmios, a literatura correlata ao PRR, outros problemas similares e aplicações. No Capítulo 3, apresenta-se a formulação matemática proposta, onde fazemos uma comparação com outras formulações existentes na literatura para o PRR adaptadas ao PRR-CP. No Capítulo 4, apresenta-se uma análise das regras de redução existentes na literatura para o PRR, assim como se propõe um conjunto de regras para a redução do grafo associado ao PRR-CP. No Capítulo 5, apresenta-se uma breve descrição das metaheurísticas GRASP (*Greedy Randomized Adaptive Search Procedure*), VNS (*Variable Neighbourhood Search*), bem como reconexão de caminhos. Além disso, propõe-se um procedimento para a construção de uma solução inicial, e procedimentos para a busca local associado ao PRR-CP, assim como as heurísticas GRASP formadas por estes procedimentos. No Capítulo 6, apresentam-se os resultados computacionais obtidos. No Capítulo 7, são apresentadas as conclusões e as propostas para a continuação do trabalho.

Capítulo 2

O Problema de Recobrimento de Rotas com Coleta de Prêmios (PRR-CP)

2.1 Descrição do Problema

O PRR-CP é uma variação do Problema de Recobrimento de Rotas (PRR), que por sua vez é uma generalização do Problema do Caixeiro Viajante (PCV). O objetivo do PRR é encontrar, em um grafo não direcionado $G = (V \cup W, E)$, um ciclo de menor custo, obedecendo a um conjunto de restrições, onde:

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

- Um vértice de W é considerado coberto se a distância entre este e pelo menos um vértice pertencente à rota for menor ou igual a uma distância $d \geq 0$, onde d é um dado de entrada para o problema;
- Neste problema, é suposto que os vértices de W não podem estar presentes numa solução.
- O vértice 1 representa o vértice origem ($1 \in T$);
- Existe uma matriz simétrica de distâncias $C = (c_{ij})$, com $i, j \in V \cup W$, definida sob o conjunto de arestas E, satisfazendo a desigualdade triangular.

O Problema de Recobrimento de Rotas (PRR) consiste em determinar uma rota ou um ciclo 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, utilizando, se necessário, os vértices de $V \backslash T$.

Neste trabalho, tratamos uma variante do PRR, denominada Problema de Recobrimento de Rota com Coleta de Prêmios (PRR-CP). Onde a cada vértice $v_k \in V$ associa-se um prêmio, não negativo p_k. Além das restrições do PRR, acrescenta-se uma restrição de coleta de uma quantidade de prêmios ($PRIZE$). Na figura 2.1, os números acima de cada vértice representam os prêmios associados aos mesmos. Note que somente os vértices de V possum prêmios. Observe ainda que o prêmio coletado pela rota é de 14 unidades. Enquanto o prêmio mínimo a ser coletado é de 10 unidades.

O objetivo do PRR-CP, assim como o do PRR, é encontrar uma rota de comprimento mínimo sob um subconjunto de V, que contenha todos os vértices de $T \subseteq V$, cobrindo todos vértices de W e coletando pelo menos uma quantidade mínima de prêmios, denotado $PRIZE$.

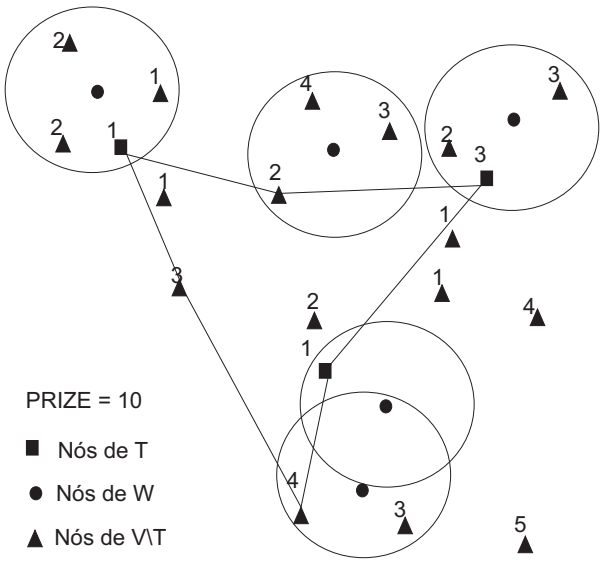

Figura 2.1: *Grafo associado à uma instância do PRR-CP.*

2.2 Literatura Existente

Nesta seção abordamos a literatura referente ao PRR, uma vez que o PRR-CP está sendo proposto neste trabalho.

Apesar da grande aplicabilidade a problemas reais, o PRR não tem recebido muita atenção na literatura desde que foi introduzido em 1981, por Current [9], que, juntamente com Schilling propõe em [13] e [11] uma heurística para gerar um conjunto de soluções para duas versões distintas do problema. A primeira versão consiste em minimizar somente o comprimento da rota, enquanto a segunda, maximiza o número de vértices cobertos pela mesma.

Gendreau, Laporte e Semet [21], apresentam e analisam a seguinte formulação matemática para o PRR.

Para cada vértice $v_k \in V$, seja y_k uma variável binária igual a 1 se e somente se o vértice v_k pertence a rota. Se $v_k \in T$ então y_k é necessariamente igual a 1. Para $v_i, v_j \in V$ e $i < j$, seja x_{ij} uma variável binária igual a 1 se e somente se a aresta (v_i, v_j) pertence a rota. Também define-se o δ_{lk} igual a 1 se e somente se $v_l \in W$ é coberto por $v_k \in V$ (i.e., $c_{lk} \leq c$) e seja $S_l = \{v_k \in V \mid \delta_{lk} = 1\}$ para todo $v_l \in W$. Onde S_l representa o conjunto dos vértices $v_k \in V$ que cobrem o vértice $v_l \in W$.

Usando as variáveis de decisão definidas acima, o PRR foi descrito em [21] da seguinte forma,

(P1)

$$minimizar \sum_{i<j} c_{ij} x_{ij}, \tag{2.1}$$

sujeito a:

$$\sum_{v_k \in S_l} y_k \geq 1 \qquad (v_l \in W), \tag{2.2}$$

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

$$\sum_{\lambda} x_{ij} \geq 2y_t \qquad (S \subset V, 2 \leq |S| \leq n-2, T\backslash S \neq \emptyset, v_t \in S), \tag{2.4}$$

$$x_{ij} \in \{0,1\} \qquad (1 \leq i < j \leq n), \tag{2.5}$$

$$y_k = 1 \qquad (v_k \in T), \tag{2.6}$$

$$y_k \in \{0,1\} \qquad (v_k \in V\backslash T), \tag{2.7}$$

Nesta formulação, as restrições (2.2) asseguram que todo vértice de W é coberto pela rota, enquanto as restrições (2.3) asseguram que todos os vértices da rota possuam, obrigatoriamente, grau 2. As restrições (2.4) são responsáveis pela conectividade da rota, forçando a precença de pelo menos duas arestas entre quaisquer cnjunto S e $V\backslash S$, para todo subconjunto próprio S de V tal que $T\backslash S \neq \emptyset$ e S contem um vértice v_t pertencente a rota. Para cada circuito hamiltoniano possível é necessário uma restrição do tipo (2.4), justificando, assim o número de $O(2^n)$ restrições. Note que, λ corresponde a ($v_i \in S$, $v_j \in V\backslash S$ ou $v_j \in S$, $v_i \in V\backslash S$), isto é, λ corresponde às arestas (i,j) que possuem uma de suas extremidades no conjunto S e a outra no conjunto $(V\backslash S)$. $|S|$ representa o número de vértices em S. Finalmente, as restrições (2.5), (2.6) e (2.7) referem-se às condições de integralidade do problema. Note que as restrições (2.6) determinam que $\forall k \in T$, $y_k = 1$. Por definição, a rota $\{1,1\}$, onde 1 é o depósito, é considerada inviável. Esta formulação destaca um importante aspecto do PRR que é sua natureza altamente combinatória.

Para adaptarmos este modelo de programação linear inteira para o PRR-CP basta acrescentarmos uma nova restrição

$$\sum_{k \in V} p_k y_k \geq PRIZE \tag{2.8}$$

A restrição (2.8) garante que o prêmio mínimo, $PRIZE$, será coletado pela rota.

O maior gargalo desta formulação é o número exponencial de restrições exigidas para se evitar a formação de subrotas desconexas da origem no PRR e PRR-CP (2^n restrições).

Ainda neste mesmo trabalho[21] os autores propuzeram 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 (PCV) [20] com o algoritmo PRIMAL1 para o *Set Covering Problem* [7]. A heurística GENIUS para o PCV, 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 dos quais, a cada passo, um novo vértice é inserido até que todos façam parte da rota. A grande diferença entre as demais formas de inserção de vértices em subrotas, consiste em não necessariamente realizar a insersão entre dois vértices consecutivos da subrota. Cada inserção é executada simultaneamente com uma reotimização local da rota corrente. Após esta fase de inserção, inicia-se a fase de reotimização, conhecida por US (*Unstringing and Stringing*). Nesta reotimização, cada vértice pode ser sucessivamente removido e reinserido na rota, usando o mesmo princípio da fase de inserção, até que nenhuma melhora possa ser obtida.

A outra heurística utilizada, PRIMAL1, inclui gradualmente na solução, variáveis de acordo com um critério "guloso", com o objetivo de minimizar de uma função $f(c_k, b_k)$, onde a cada passo, b_k representa o número de vértices $v_l \in W$, com $\delta_{lk} = 1$, ainda não cobertos pela rota, onde δ_{ij} é um coeficiente binário que é igual a 1, se e somente se, $i \in W$ é coberto por $j \in V$. Os três critérios para inclusão de vértices utilizados na heurística foram sugeridos por Balas e Ho para o *Set Covering Problem* [7] são:

(i) $f(c_k, b_k) = c_k / log b_k$,

(ii) $f(c_k, b_k) = c_k/b_k$ e

(iii) $f(c_k, b_k) = c_k$.

A heurística PRIMAL1 primeiramente aplica o critério (i) de maneira gulosa até que todos os vértices de W estejam cobertos. De acordo com a ordem de inclusão, as variáveis y_k são removidas da solução quando δ_{lk} é igual a 1 para um vértice $v_l \in W$ coberto por mais de um v_k. Então, a cobertura parcial é feita segundo o critério (ii). Novamente, as variáveis y_k que cobrem um mesmo v_l mais de uma vez são removidas e o critério (iii) é aplicado a cobertura parcial. Uma segunda solução é construída aplicando-se o conjunto de critérios na seguinte ordem (i), (iii) e (ii). A melhor das duas soluções é armazenada.

A heurística apresentada por Gendreau *et al.* [21] constrói a rota iterativamente inserindo os vértices na subrota existente, utilizando a heurística GENIUS. A seleção dos vértices a serem inseridos a cada iteração é feita de acordo com o critério guloso corrente, 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. Para tanto é feita 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 ou dominate seja obtida ou até que seja mais promissor particionar o espaço de busca.

Os testes são realizados com alguns problemas onde somente um subconjunto de vértices são visitados. Dentre os quais, estão o *Prize Collecting Travelling Salesman Problem* (PCTSP) [5], o *Selective Travelling Salesman Problem* (STSP) [34] e o *Generalized Travelling Salesman Problem* (GTSP) [18]. Os algoritmos também são testados com uma série de problemas gerados aleatoriamente, onde os $|V| + |W|$ vértices são gerados 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 coeficien-

tes c_{ij} (custo associado à aresta (i, j)) são computados como a distância Euclideana entre os vértices i e j. Os algoritmos são implementados em C e executados em uma estação SunSPARC 1000. Testes forão realizados com instâncias onde $n = 50$, 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). As que envolviam 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. Como conseqüência, a árvore de busca resultante é relativamente pequena, entretanto, um certo esforço computacional é despendido em cada vértice. Infelizmente os problemas testes usados pelos autores não se encontraram disponíveis em bibliotecas públicas.

Em 1999, Maniezzo *et al.* [35] apresentaram três algoritmos utilizando a metaheurística *Scatter Search* proposta por Glover [23]. A idéia por trás desta técnica é o uso de um conjunto R de soluções, ou pontos representativos, chamado *Conjunto Referência*. Operadores de recombinação são aplicados nos pontos de R, para gerar novas soluções. Cada uma destas será usada como entrada para procedimentos de busca local com o intuito de se obter soluções melhores. Combinações lineares são os meios utilizados para se combinar as soluções. As melhores soluções obtidas ao final deste processo de recombinação são inseridas no conjunto referência. Todo o processo é repetido por um determinado número de iterações.

O primeiro algoritmo apresentado por Maniezzo *et al.* em [35], denotado por **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 [22], diferindo dos dois anteriores por não usar cortes. Os três algoritmos compartilham uma série de procedimentos de problemas relacionados ao PRR.

Nos testes computacionais são utilizados os quatro testes para a redução do grafo original, apresentados em [21]. 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 [8] é utilizado para resolver o problema Lagrangeano. As instâncias são geradas como propostas por Gendreau et al. [21], 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. [21], constatando que a heurística de Gendreau é muito mais rápida que o *Scatter Search*, proposto em [35].

Neste mesmo artigo, Maniezzo *et al.* [35] propõem uma formulação matemática para o PRR. Este modelo foi proposto, originalmente, por Finke, Clauss e Gunn [17] para o TSP e adaptado para o PRR.

Tal formulação 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. Se a rota vai do vértice i ao j, então x_{ij} representa o número de vértices que podem ser visitados e x_{ji} representa o número de vértices já visitados. Assim, a variável x_{ij} define dois circuitos para qualquer rota que representar uma solução viável. Um circuito é definido pelas variáveis de fluxo representando o número de vértices que podem ser visitados, enquanto o segundo circuito é definido pelas variáveis de fluxo representando o número de vértices já visitados. E finalmente, temos duas variáveis binárias, ξ_{ij} é igual a 1 se, e somente se, a aresta $(i, j) \in E$ está na solução e zero, caso contrário e y_i é igual a 1 se, e somente se, o vértice $i \in V$ for visitado e zero, caso contrário.

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

(P2)

$$minimizar \sum_{(i,j)\in E} c_{ij}\xi_{ij}, \tag{2.9}$$

sujeito a:

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = -2y_i \qquad (i \in V), \tag{2.10}$$

$$\sum_{j\in V}(x_{ij} + x_{ji}) = 2y_i(|V| - 1), \qquad (i \in V) \tag{2.11}$$

$$\sum_{i\in G_h} y_i \geq 1, \qquad (G_h = \{i \in V \ : \ c_{ih} \leq d, h \in W\}) \tag{2.12}$$

$$x_{ij} + x_{ji} = (n-1)\xi_{ij}, \qquad (i,j) \in E \tag{2.13}$$

$$x_{ij} \geq 0, \qquad i,j \in V,\ i \neq j \tag{2.14}$$

$$y_j = 1 \qquad (j \in T) \tag{2.15}$$

$$y_i \in \{0,1\} \qquad (j \in V \backslash T) \tag{2.16}$$

$$\xi_{ij} \in \{0,1\} \qquad (i,j) \in E \tag{2.17}$$

Nesta formulação, as restrições (2.10) e (2.14) definem um fluxo viável para as variáveis x_{ij}. As restrições (2.11) e (2.13) garantem que todo vértice i na rota possui grau 2. As restrições (2.12) asseguram a cobertura de todos os vértices de W, garantindo que pelo menos um vértice $i \in V$, tal que $c_{ih} \leq d$ esteja na rota. As restrições (2.15) garantem que todos os vértices $i \in T$ pertencem à rota. Finalmente, as restrições (2.16) e (2.17) referem-se as condições de integralidade do problema.

Observe que, assim como em $(P1)$, para adaptarmos este modelo para o PRR-CP, basta acrescentarmos a restrição (2.8).

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

2.3 Problemas Similares e Aplicações

Existem na literatura diversos problemas similares ao PRR. Entre eles estão: o *Shortest Covering Path Problem* (SCPP) [11, 10], o *Multi-Vehicle Covering Tour Problem* (m-CTP) [27], o *Median Tour Problem* (MTP)[12], o *Maximal Covering Tour Problem* (MCTP) [12], entre outros.

O SCPP foi formulado em 1984 por Current *et al.* [10] e é definido sob um

grafo completo e direcionado $G = (N, A)$, 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 retornando à 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. 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. O m-CTP consiste em determinar um conjunto de m rotas de comprimento mínimo, de modo que as rotas satisfaçam às seguintes restrições:

1. Todas as m rotas de comprimento mínimo, devem partir e retornar ao depósito s ($s \in T$) previamente estabelecido;

2. Cada vértice de V deve pertencer no máximo a uma rota, enquanto cada vértice de $T \subseteq V$ deve pertencer 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 vértice $i \in W$);

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 $m = 1$ e o valor pré-fixado p for grande o suficiente para conter, no pior caso, todos os vértices de V, assim como 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, 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 [32] .

O *Prize Collecting Travelling Salesman Problem* (PCTSP) [6, 19] e o *Selective Travelling Salesman Problem* (STSP) [34] também podem ser vistos como problemas similares ao PRR e ao PRR-CP. 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 PTCSP consiste na minimização do custo da rota, onde o total de prêmios coletados é no mínimo p. Alem disso, a cada vértice não pertencente à rota, existe uma penalidade associada. O STSP, por sua vez, consiste na maximização do total de prêmios coletados pela rota cujo comprimento total não deve exceder um certo valor r, pré-fixado.

Pode-se ainda citar o Median Tour Problem (MTP) e o Maximal Covering Tour Problem (MCTP) [12], que se assemelham em muitos pontos ao PRR. 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 [39], 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) [4], onde o conjunto de vértices não é um conjunto discreto, mas sim uma 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 [38] ou ainda na identificação aérea de focos de incêndio em florestas [31].

Existem ainda muitas outras aplicações para o PRR. 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 *Travelling Circus Problem* [41] e consiste em encontrar uma rota onde as localidades não incluídas no planejamento de paradas estejam acessíveis a pelo menos uma das incluídas. Para cada localidade não visitada acrescenta-se uma penalidade.

Uma variante do PRR, denominada Problema de Recobrimento de Rotas Generalizado (PRRG), foi proposta por Motta em [36]. Esta generalização difere do PRR, no sentido em que permite que os vértices do conjunto W façam parte da solução. É apresentada uma formulação matemática, as regras de redução existentes na literatura para o PRR são analisadas e foi mostrado ainda que duas destas regras podem gerar soluções iniviáveis. Neste trabalho, também é apresentado um conjunto de regras para a redução do grafo associado ao PRRG e são propostas heurísticas para a solução aproximada do problema.

Aplicações do PRR-CP incluem alguns problemas de roteamento de veículos coletores de derivados de petróleo em poços terrestres semi desativados, onde em cada vértice (poço) de V existe associado determinadas quantidades de produtos (óleo) a serem coletadas. Normalmente uma solução contém apenas parte dos vértices de V devido à capacidade limitada do veículo coletor [14]. Outras aplicações envolvem

modelos de despacho de mercadorias onde um veículo de maior porte entrega os produtos em bases maiores (depósitos), por exemplo, vértices de T e alguns vértices de $V \backslash T$ e os vértices de W (clientes menores) devem ser abastecidas por uma destas bases utilizando veículos menores, mas com a condição dos pontos de W estarem a uma distância menor ou igual a uma distância máxima prefixada de pelo menos uma das bases. Neste caso, o prêmio poderia ser associado à capacidade dos depósitos ou à soma das demandas dos clientes a ele associados.

Capítulo 3

Formulação Matemática para o PRR-CP

Nesta seção, apresentamos um modelo de Prgramação Linear Inteira para o PRR-CP, utilizando variáveis de fluxo, com o intuito de evitar a formação de ciclos desconexos da origem.

3.1 Formulação proposta

Nesta seção propomos uma nova formulação matemática para o PRR-CP. Nela introduzimos variáveis de fluxo com a finalidade de evitar a formação de ciclos desconexos da origem. Para cada aresta (i, j) associamos uma variável de fluxo, z_{ij} com $i \neq j$, uma variável inteira não negativa, que representa a quantidade de fluxo escoado no arco (i, j). Além disso, temos y_k, com $k \in V$, variável binária, igual a 1 se, e somente se, o vértice k estiver na rota e zero, caso contrário. O vértice $s = 1$ é o vértice origem. Além disso, x_{ij}, $\forall i, j \in V$ e $i \neq j$, é uma variável binária, igual a 1 se, e somente se, a aresta (i, j) pertence à rota e 0 (zero), caso contrário. Finalmente, c_{ij} representa o custo de se ir de i para j; p_k, o prêmio associado a cada vértice $k \in V$; e $PRIZE$, o prêmio mínimo a ser coletado pela rota.

Assim, esta nova formulação pode ser escrita como um problema de programação

linear inteira da seguinte forma:

(P3)

$$minimizar \sum_{i<j|i,j\in V} c_{ij}x_{ij}, \tag{3.1}$$

sujeito a:

$$\sum_{k\in V} p_k y_k \geq PRIZE, \tag{3.2}$$

$$\sum_{k\in S_l} y_k \geq 1 \qquad (\forall l \in W), \tag{3.3}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2y_k \qquad (\forall k \in V), \tag{3.4}$$

$$\sum_{j\in V} z_{kj} = \sum_{i\in V} z_{ik} + y_k \qquad (\forall k \in V - \{s\}), \tag{3.5}$$

$$\sum_{j\in V} z_{sj} = 1, \tag{3.6}$$

$$\sum_{j\in V} z_{js} = \sum_{j\in V} y_j, \tag{3.7}$$

$$x_{ij} \leq z_{ij} \qquad (\forall i\ \forall j \in V), \tag{3.8}$$

$$x_{ij} \geq z_{ij}/(|V|+1) \qquad (\forall i\ \forall j \in V), \tag{3.9}$$

$$y_k = 1 \qquad (\forall k \in T), \tag{3.10}$$

$$y_k \in \{0,1\} \qquad (\forall k \in V\backslash T), \tag{3.11}$$

$$x_{ij} \in \{0,1\} \qquad (\forall i\ \forall j \in V), \tag{3.12}$$

$$z_{ij} \in \mathbb{Z} \qquad (\forall i\ \forall j \in V), \tag{3.13}$$

Em $(P3)$, a função objetivo (3.1) minimiza o custo da rota. A restrição (3.2) garante que o prêmio mínimo, $PRIZE$, será coletado. Note que, retirando-se (3.2) obtemos uma formulação para o PRR. As restrições (3.3) asseguram que cada vértice de W seja coberto por pelo menos um vértice da rota. O conjunto $S_l = \{k \in V\backslash c_{lk} \leq d, l \in W\}$ representa o conjunto de todos os vértices de V que cobrem $l \in W$. As restrições (3.4) restringem a dois o grau dos vértices da rota. As restrições de (3.5) a (3.7) asseguram a não formação de ciclos desconexos da origem. As restrições (3.8) e (3.9) 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 (3.10) garantem que todos

os vértices de T fazem parte da rota. Finalmente, as restrições de (3.11) a (3.13) representam as condições de integralidade do problema.

3.1.1 Comparação entre as três formulações

Nesta seção, mostramos uma comparação entre os três modelos descritos em $(P1)$, $(P2)$ e $(P3)$, os dois primeiros são modelos já existentes na literatura para o PRR e adaptados aqui para o PRR-CP, e o último é o modelo aqui proposto. Na tabela 3.1, a primeira coluna mostra o modelo que estamo tratando, na segunda coluna temos o número de restrições e na terceira coluna, o número de variáveis originais de cada modelo. Onde $n = |V \cup W|$, $p = |V|$ e $m = |E|$, onde E é o conjunto de arestas do grafo de entrada. Note que, para um grafo completo $m = \frac{n(n-1)}{2}$. Posteriormente, apresentamos os resultados computacionais obtidos com a nossa formulação $(P3)$.

	restrições	variáveis
$(P1)$	$O(2^n)$	$m + n + p(n - p)$
$(P2)$	$O(n^2)$	$3m + p$
$(P3)$	$O(n^2)$	$2m + p$

Tabela 3.1: Comparação entre os modelos pelo número de restrições e variáveis exigidas

Capítulo 4

Regras de Redução para o PRR e o PRR-CP

Um dos principais gargalos dos problemas de Otimização Combinatória é justamente o tamanho de suas entradas. Se conseguirmos reduzir o espaço de busca dos problemas desta natureza também reduzimos o esforço computacional para se chegar a um ótimo global. É este o objetivo das regras de redução, reduzir o espaço de busca sem excluir nenhuma solução ótima.

Neste capítulo apresentamos e analisamos regras de redução para o PRR existentes na literatura e as adaptações destas para o PRR-CP, bem como, uma nova regra proposta para redução do grafo associado.

4.1 Regras de Redução para o PRR

As regras para a redução do espaço de busca descritas a seguir foram introduzidas por Gendreu, Laporte e Semet [21], e posteriormente utilizadas por Maniezzo et al [35] para o PRR. Considere a variável binária, ξ_{ij}, 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 $\xi_{ij} = 0$ (zero), caso contrário. O conjunto de regras de redução apresentadas em [21] é descrito por:

1. Remover $i \in W$, se $\forall j \in V \backslash T$, $\xi_{ij} = 1$;
2. Remover $i \in W$, se $\exists j \neq i$, com $j \in W$, tq $\xi_{ik} \leq \xi_{jk}$, $\forall k \in V \backslash T$;
3. Remover $i \in W$, se $\exists j \in T$, tq j cubra i, isto é, $\xi_{ij} = 1$;
4. Remover $i \in V \backslash T$, se $\forall j \in W, \xi_{ij} = 0$.

Motta [36] mostra que as regras (1) e (2) podem gerar soluções inviáveis para o PRR. Com relação às regras (3) e (4), estas se aplicam perfeitamente ao PRR. Neste trabalho adaptamos o conjunto de regras anterior para o PRR com Coleta de Prêmios (PRR-CP). Propomos uma correção para a regra (1), bem como, uma nova regra. A regras (3) pode ser aplicada e a regra (4) com uma pequena modificação poderá ser utilizada. Ainda, mostramos que se utilizarmos estas regras em uma determinada ordem, o processo de redução do grafo será otimizado ainda mais.

Considere o grafo da Figura 4.1, para análise da regra (1) . Neste grafo, todo vértice de $V \backslash T$ cobre todo vértice W. Entretanto, ao remover os vértices de W, não necessitamos mais dos vértices de $V \backslash T$ na solução. Logo, uma solução viável para o grafo reduzido teria somente os vértices de T. Portanto, qualquer algoritmo encontrará uma solução inviável associada ao grafo original, como ilustrado na Figura 4.2.

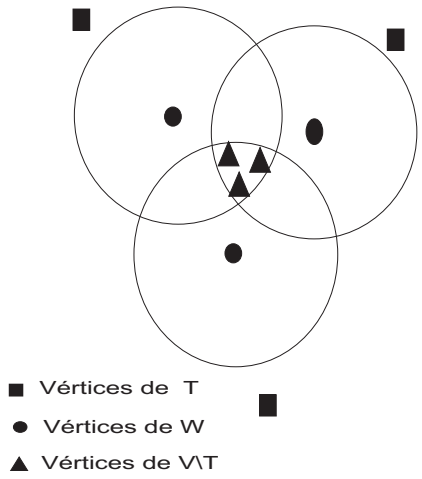

Figura 4.1: *Grafo associado a uma instância do PRR, para análise da regra (1).*

Em relação à regra (2), esta elimina do grafo original todo vértice $i \in W$, quando existir um outro vértice $j \neq i$ em W, tal que todo $k \in V \backslash T$ pertencente à vizinhança

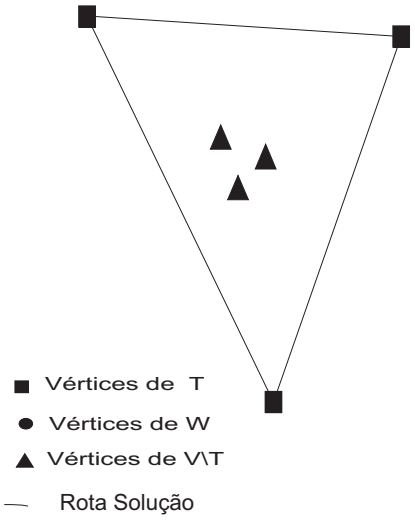

Figura 4.2: *Grafo da Figura 4.1, após a redução, com a rota ótima (inviável para o grafo original).*

de i, também pertence à vizinhança de j, mas nem todo $k \in V \backslash T$ pertencente à vizinhança de j, pertence à vizinhança de i. Observe a situação ilustrada na Figura 4.3. Realizando a redução segundo a regra (2), um vértice de W será eliminado (Figura 4.4) e o grafo reduzido terá como solução ótima (Figura 4.5), uma solução inviável ao grafo original.

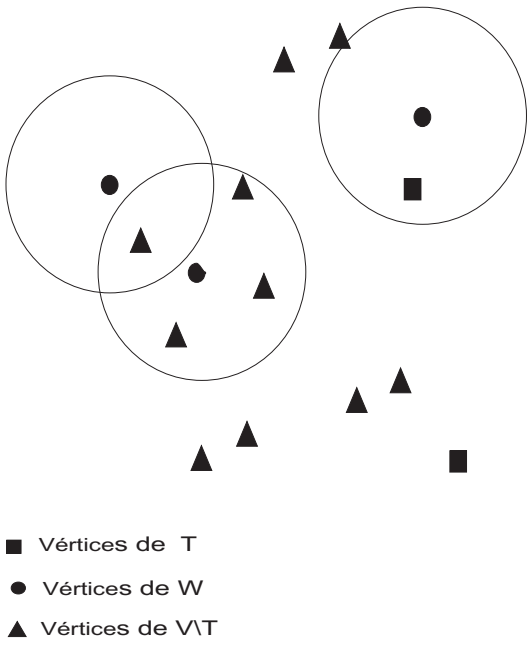

Figura 4.3: *Grafo associado a uma instância do PRR para análise da regra (2)*

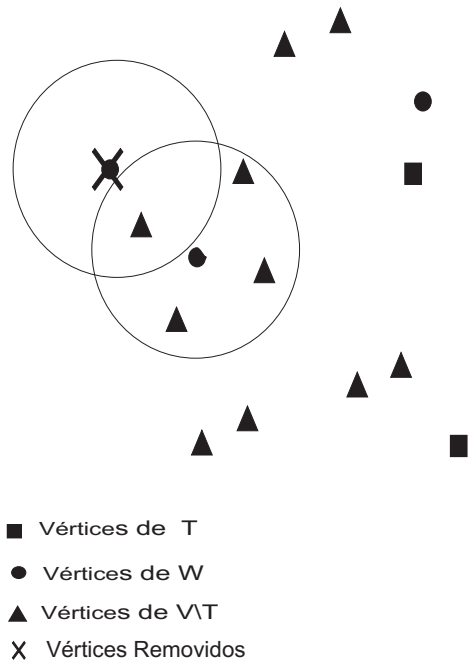

Figura 4.4: *Redução do grafo da Figura 4.3 pela regra (2)*

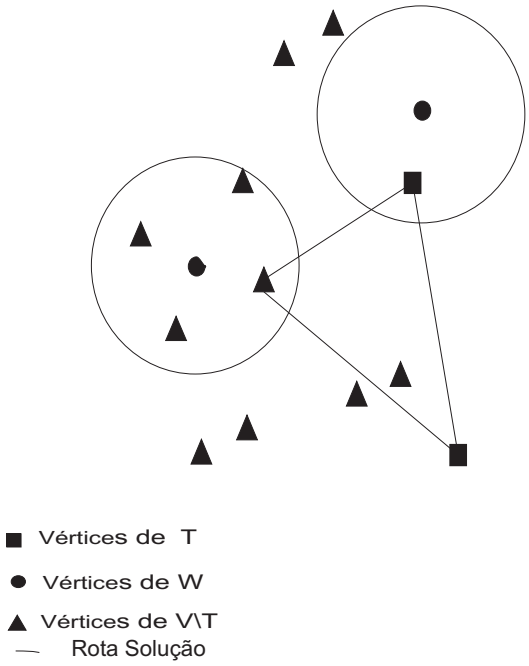

Figura 4.5: *Grafo da Figura 4.3 após a redução, pela regra (2). A rota associada é inviável no grafo original.*

Note que, na Figura 4.5, que a rota ótima associada ao grafo reduzido não é viável no grafo original. Com relação às regras (3) e (4), estas se aplicam perfeitamente ao PRR.

Veja nas Figuras 4.6 e 4.7 a interpretação da regra (3), que retira do grafo os nós de W que forem cobertos por algum nó de T.

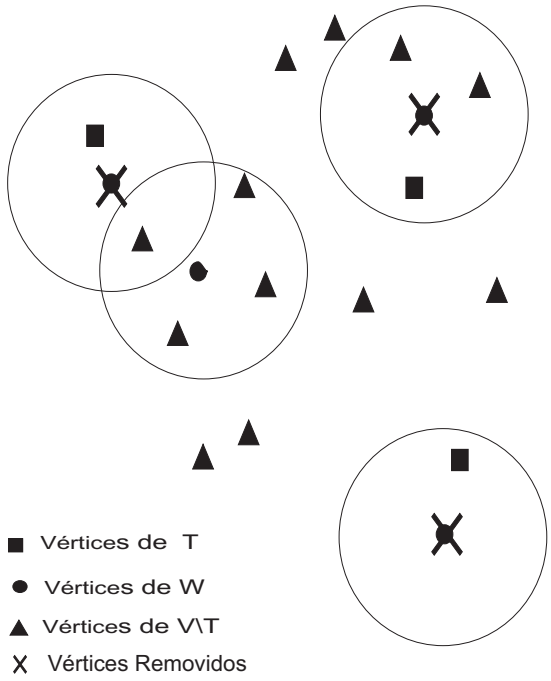

Figura 4.6: *Instância do PRR, após análise da regra (3).*

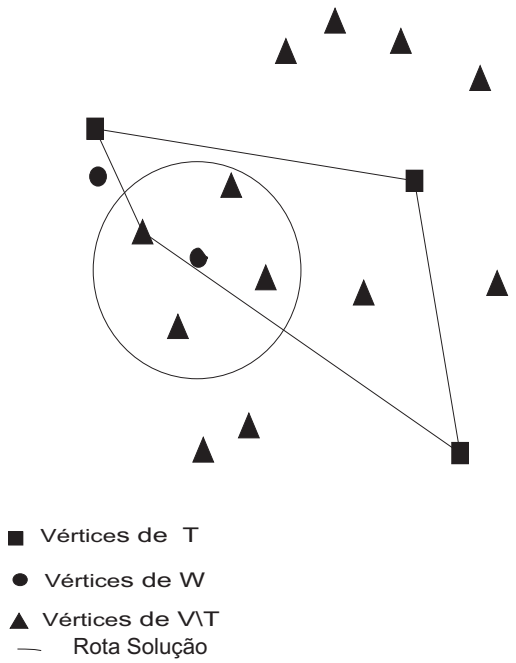

Figura 4.7: *Grafo da Figura 4.6, após redução pela regra (3), com a rota ótima associada.*

E ainda, segundo a regra (4), podemos eliminar os vértices de $V\backslash T$ que não cubram nenhum vértice de W, ver Figura 4.8.

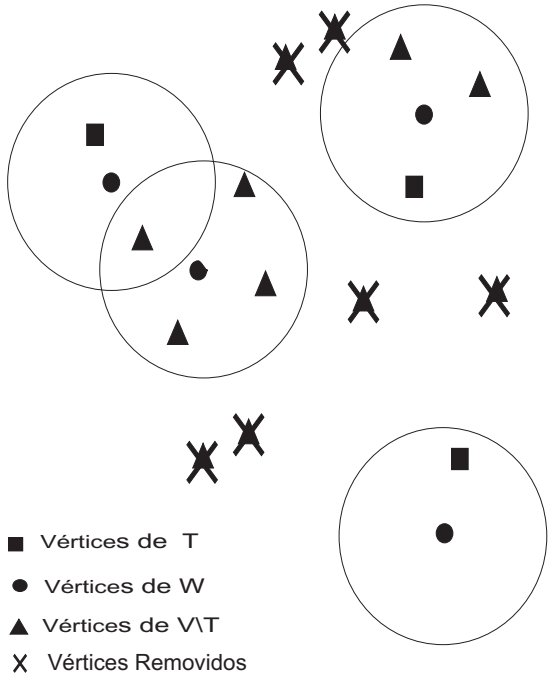

Figura 4.8: *Grafo da Figura 4.6, após redução pela regra (4).*

4.1.1 Análise das Regras existentes associadas ao PRR-CP

Mostramos nesta seção, como adaptar o conjunto de regras existentes para o PRR ao PRR com Coleta de Prêmios (PRR-CP). As regras (1) e (3) podem ser aplicadas e a regra (4) com uma pequena modificação poderá ser utilizada. Além disso, também mostramos que se utilizarmos estas regras em uma determinada ordem, o processo de redução do grafo pode ser otimizado ainda mais. Vamos reescrever as regras com uma nova numeração, adaptar a regra (4) do PRR e propor uma nova regra para o PRR-CP.

R1. Remover $i \in W$, se $\forall j \in V\backslash T$, $\xi_{ij} = 1$ e setar $flag = 1$;

R2. Remover $i \in W$, se $\exists j \in T$, tq j cubra i, isto é, $\xi_{ij} = 1$;

R3. Se $\sum_{i \in T} p_i \geq PRIZE$, remover $i \in V\backslash T$, se $\forall j \in W, \xi_{ij} = 0$.

R4. Se apenas um $j \in V$, cobre um $i \in W$ ($\xi_{ij} = 1$), transforme o vértice j em vértice de T.

As regras (R1) e (R2) são as regras (1) e (2) da literatura. Note que, caso a regra (R1) seja aplicada e elimine algum elemento de W teremos que utilizar uma variável auxiliar, *flag*, para sabermos que algum vértice de $V\backslash T$ terá que entrar na solução. A regra (R3) precisou ser modificada, pois cada vértice do conjunto V terá um prêmio associado, e $v_k \in V$ só será descartado caso este não seja necessário para que o prêmio mínimo ($PRIZE$) seja obtido. Isto é, quando os vértices de T já atendam a esta restrição. Propomos ainda a reagra (R4) que reduz a cardinalidade do conjunto $V\backslash T$. Veremos s seguir porque a aplicação desta regra é interessante para o problema.

A primeira regra a ser aplicada deverá ser (R4). Esta irá trasformar um vértice j optativo ($j \in V\backslash T$) em vértice obrigatório. Caso somente este j possa ser usado para cobrir algum vértice $i \in W$. Esta regra poderá reduzir a cardinalidade de $V\backslash T$ (o que ajudará na aplicação da regra (R1)) e consequentemente irá aumentar a cardinalidade de T (o que ajudará na remoção de vértices por (R2) e (R3))

Em seguida, devemos aplicar a regra (R3) modificada para reduzir, se for o caso, a cardinalidade do conjunto $V\backslash T$. O que, mais tarde, ajudará na reução do grafo pela regra (R1), como veremos a seguir (Figura 4.11).

Depois de (R3), vamos aplicar a regra (R2). Neste caso, podemos remover todo $i \in W$, onde i está coberto por $j \in T$. Caso todo $i \in W$ esteja coberto por algum $j \in T$, então teremos uma situação ideal e não vamos nos preocupar com a condição de cobertura destes vértices. Caso contrário, algum vértice $v \in V\backslash T$ terá que entrar na solução e é neste ponto que se enquadra (R1). Esta regra, que até então não era utilizada, com o recurso do *flag* (variável auxiliar) poderá ajudar na redução do grafo. Ela vai remover $i \in W$, se i for coberto por todo $v \in V\backslash T$.

A regra (R3) dever ser aplicada antes de (R1), pois poderá reduzir o conjunto $V\backslash T$, o que irá ajudar na aplicabilidade desta regra.

Caso a regra (R2) seja aplicada antes da (R1), pode ser que ela exclua todos os vértices de W então (R1) não será utilizada, pois $|W| = 0$. Caso contrário, se após aplicarmos a regra (R2) e o conjunto W ainda não estiver vazio, então vamos aplicar a regra (R1). Feito isso, caso o conjunto W esteja vazio, nós saberemos que será necessário se incluir um vértice de $V \backslash T$ na rota para que nenhum nó de W, do grafo original, fique descoberto. Vejamos como a ordem de aplicação das regras pode influenciar na redução do grafo. Suponha que a regra (R1) foi utilizada antes da (R3). Para o exemplo da Figura 4.9, suponhamos que o prêmio mínimo esteja atendido pelos vértices de T.

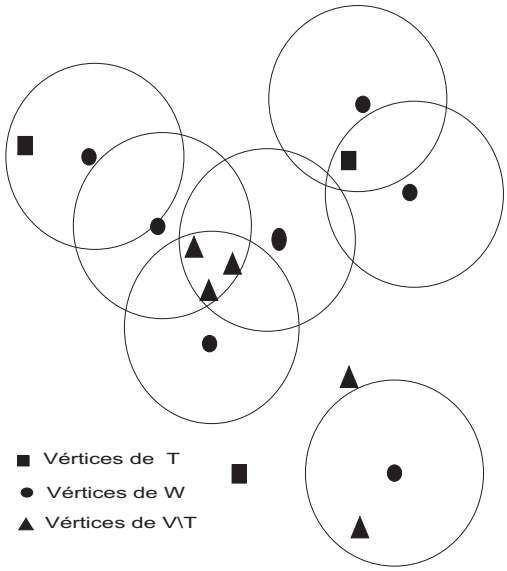

Figura 4.9: *Grafo associado a uma instância do PRR-CP*

Se aplicássemos a regra (R1), ao grafo da Figura 4.9, ela não eliminaria nenhum vértice de W, pois nem todos os vértices de $V \backslash T$ cobrem algum vértice de W. Contudo, aplicando a regra (R3) obtemos o grafo da Figura 4.11. Assim, se aplicarmos a regra (R1), ao grafo da Figura 4.11 poderemos eliminar os vértices de W que são cobertos por todos os $v \in V \backslash T$. Além disso, é interessante aplicarmos (R2) antes de (R1) uma vez que (R2) pode tornar conjunto W vazio então a regra (R1) será desnecessária. Caso contrário, saberemos que será preciso acrescentar um elemento de $V \backslash T$ na rota para que a solução seja válida no grafo original. Vejamos, então, na Figura 4.12 a redução do grafo pela regra (R2).

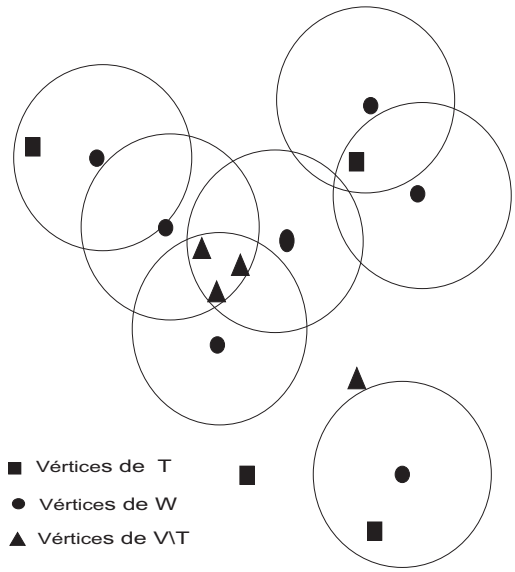

Figura 4.10: *Grafo associado a uma instância do PRR-CP, após aplicação da regra (R4)*

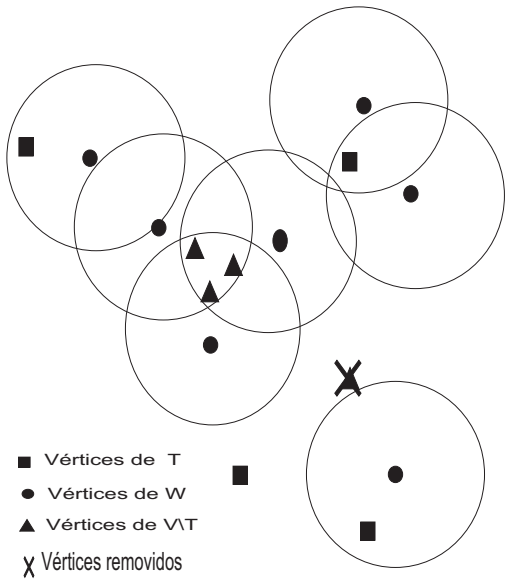

Figura 4.11: *Grafo associado a uma instância do PRR-CP, após a redução pela regra (R3)*

Na Figura 4.13, após a redução pela regra (R1) obtemos $|W| = 0$, sabemos então, que para se obter a rota ótima associada ao grafo original teremos que incluir um vértice de $V \backslash T$ do grafo reduzido.

A regra (R3) só terá efeito se a restrição do prêmio mínimo for atendida pelos vértices de T. As regras (R1) e (R2) podem ser aplicadas de forma incondicional,

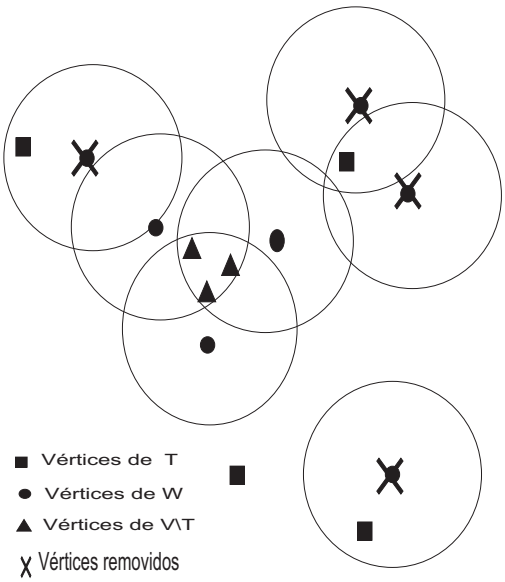

Figura 4.12: *Grafo associado ao PRR-CP, após a redução pela regra (R2)*

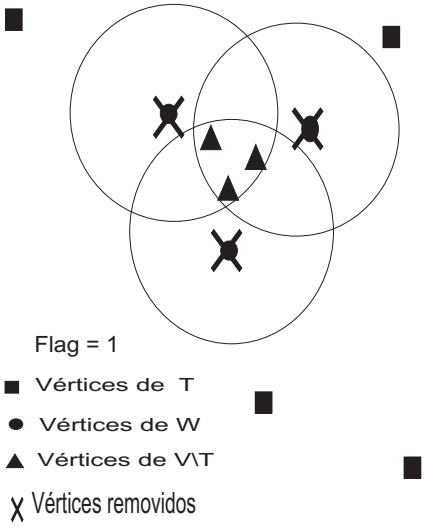

Figura 4.13: *Grafo associado ao PRR-CP, após a redução pela regra (R1)*

pois os vértices de W não possuem prêmios e também não podem entrar na rota.

Vejamos agora, regras para eliminar vértices de $V\backslash T$ quando estes vértices forem necessários para atender a restrição do prêmio mínimo e todo o conjunto W já estiver coberto. Para isso vamos estudar dois casos.

Caso (1): existe um $v_i \in V\backslash T$ cuja inclusão atende ao Prêmio Mínimo e o seu custo de inserção na rota corrente menor que o custo de inserção dos demais v_k de $V\backslash T$ que atendem a mesma restrição. Então, vamos eliminar todos os $v_k \in V\backslash T$,

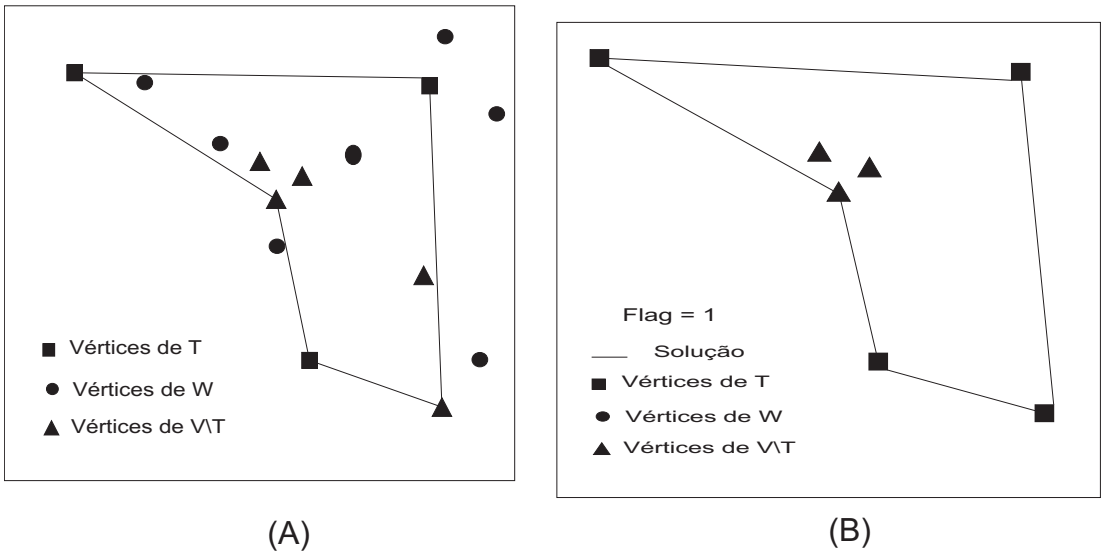

Figura 4.14: *(A) Grafo Original, com rota ótima associada. (B) Grafo Reduzido, com rota ótima associada.*

tal que $v_i \neq v_k$. No exemplo da Figura 4.15.a, temos todo $w_i \in W$ coberto e três vértices de $V \backslash T$. Suponhamos que associado a cada v_k esteja um prêmio maior ou igual à quantia que falta para completar o *PRIZE* (prêmio mínimo). Assim, serão eliminados todos os $v_k \in V \backslash T$, exceto aquele cuja inclusão seja mínima (Figura 4.15.b).

Caso(2): Este é uma generalização do caso anterior. Onde existem k vértices de $V \backslash T$ que atendem ao prêmio mínimo. Seja Δ a quantidade de prêmios que falta ser coletada na atual solução para atingir a quantidade *PRIZE* e,

$$C_i = \{v_j \in V \backslash T; \sum_{v_j \in C_i} p_j \geq \Delta\}$$

Neste caso, vamos verificar para qual destes conjuntos o custo da inclusão será mínima. E excluiremos os demais conjuntos.

Em cada conjunto C_i a soma de seus prêmios atende a diferença (Δ). Porém, o melhor conjunto, o de custo mínimo, para inclusão é o C_3. Portanto, os demais serão excluídos (Figura 4.15.d).

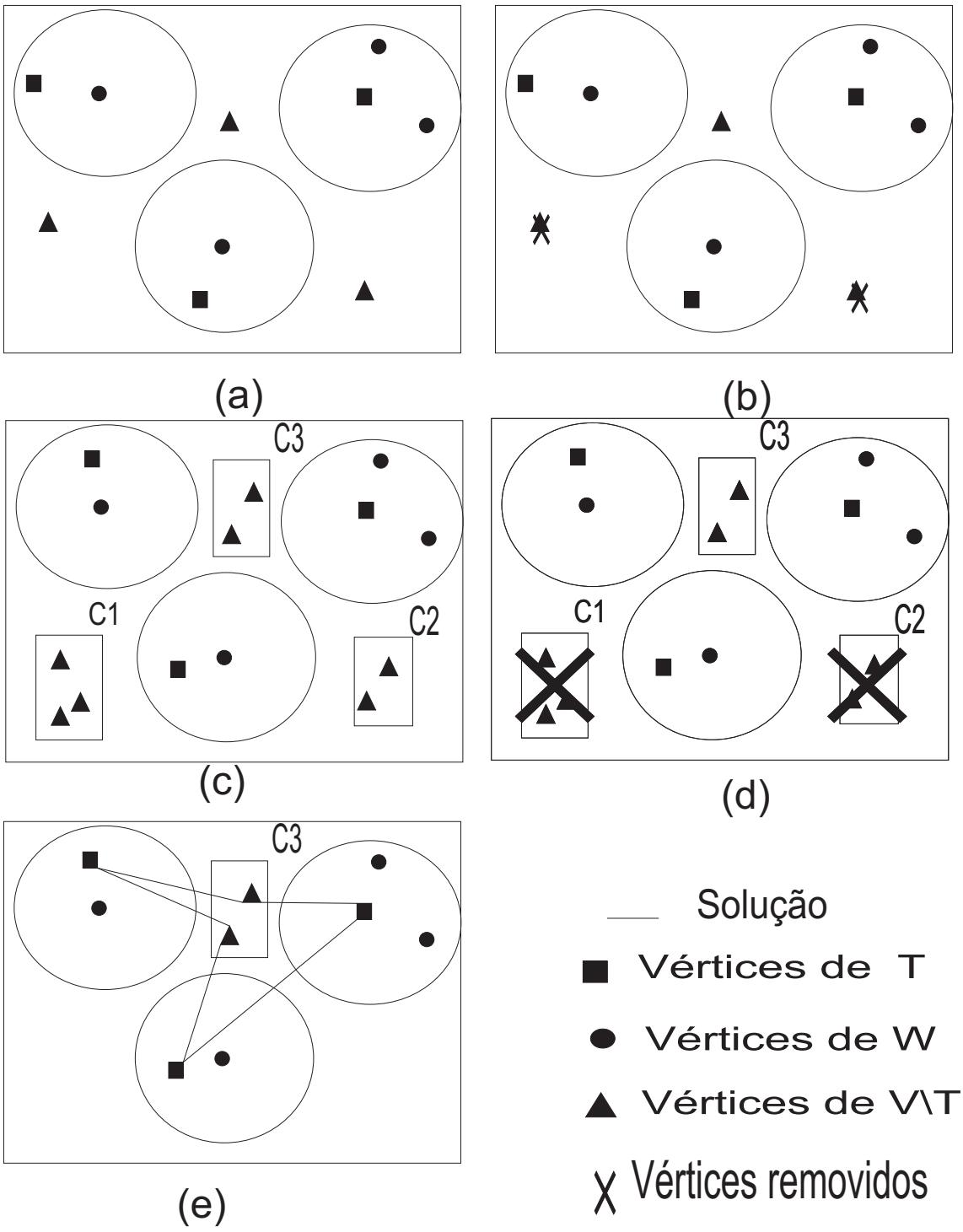

Figura 4.15: (a) *Grafo associado a uma instância do PRR-CP, para análise do* ***Caso(1)***. (b) *Redução do grafo da Figura 4.15.a.* (c) *Grafo associado a uma instância do PRR-CP para análise do* ***Caso(2)***. (d) *Redução do grafo da Figura 4.15.c* (e) *Rota associada a instância da Figura 4.15.c*

Capítulo 5

Heurísticas para o PRR-CP

5.1 Introdução

O método GRASP (*Greedy Randomized Adaptive Search Procedures*) [40] é um procedimento iterativo onde cada iteração é composta de uma fase de construção, na qual uma solução viável é construída, e de uma fase de busca local, cujo objetivo é encontrar um ótimo local. Ambas as fases são descritas a seguir.

Na fase construtiva, uma solução viável é construída também iterativamente, inserindo-se na solução parcial um elemento de cada vez. A cada iteração da fase construtiva, são avaliados apenas elementos que podem ser adicionados à solução sem violar as restrições de viabilidade. Esses elementos são chamados *elementos candidatos*. A escolha do próximo elemento a ser adicionado à solução é determinada ordenando-se todos os elementos candidatos em uma lista de candidatos C, de acordo com uma função gulosa $g : C \rightarrow \mathbb{R}$. Essa função mede o benefício associado à seleção de cada elemento. A heurística é adaptativa porque os benefícios associados a cada elemento são atualizados a cada iteração da fase construtiva, para incorporar as mudanças causadas pela escolha dos elementos anteriores. A componente probabilística é caracterizada pela escolha aleatória de um dos melhores candidatos de C, que não necessariamente o melhor. A lista dos melhores candidatos é denominada de *Lista de Candidatos Restrita* (LCR), que difere de C pois LCR contem apenas

os melhores elementos de C.

A Figura 5.1 mostra o pseudo-código do procedimento semi-guloso usado na fase construtiva.

```
1.   procedimento CONSTRUTIVO(semente)
2.     x = ∅;
3.     Inicialize o conjunto de candidatos C;
4.     Enquanto C ≠ ∅ faça
5.       Construa LCR;
6.       i = rand(semente);
7.       Selecione s_i de LCR;
8.       x = x ∪ {s_i};
9.       Atualize o conjunto de candidatos C;
10.    retorne (x);
```

Figura 5.1: Algoritmo usado na fase construtiva.

Na maior parte dos casos, é possível melhorar a solução construída, aplicando-se uma busca local a ela. A fase de Busca procura melhorar s solução construída. Os algoritmos de busca local funcionam de maneira iterativa, substituindo sucessivamente a solução corrente por outra melhor em sua vizinhança. O pseudo-código de um algoritmo básico de busca local iniciado a partir de uma solução x^0 construída na fase de construção e usando-se uma vizinhança N é apresentado na figura 5.2.

```
1.  procedimento BUSCA_LOCAL(x^0)
2.    Inicialize x = x^0;
3.    Enquanto existe x' ∈ N(x) tal que f(x') < f(x) faça
4.      Selecione x' ∈ N(x) tal que f(x') < f(x);
5.      x = x';
6.    retorne (x);
```

Figura 5.2: Procedimento básico de busca local (para um problema de minimização).

A idéia básica da metaheurística GRASP consiste de um procedimento do tipo multi-partida usando diferentes soluções iniciais como pontos de partida para a busca local. Uma solução x é dita como pertencente ao *vale* de um ótimo local quando, a partir de uma busca local iniciada em x, é possível atingir este ótimo local. Caso uma das soluções iniciais esteja em um vale de um ótimo global, a busca local irá encontrar este ótimo global. Caso contrário, a solução do algoritmo será um ótimo local.

```
1.  procedimento GRASP
2.    Repita por Max_Iter
3.      Construa uma solução randômica gulosa;
4.      Use busca local para melhorar a qualidade da solução construída;
5.      Atualize a melhor solução encontrada;
```

Figura 5.3: Algoritmo GRASP básico.

Uma solução no vale de um ótimo global será eventualmente produzida, caso um número grande de soluções geradas aleatoriamente seja usado para iniciar a busca local. Porém, soluções produzidas aleatoriamente são, em geral, de baixa qualidade. Além disso, o número de movimentos necessários para que soluções geradas aleatoriamente (e que estejam no vale de um ótimo global) atinjam o ótimo possivelmente será muito elevado ou até mesmo exponencial no tamanho do problema. Por outro lado, algoritmos gulosos geralmente produzem soluções de melhor qualidade do que as geradas aleatoriamente. O uso de soluções gulosas como ponto de partida para uma busca local, em geral, levará a boas soluções, porém, sub-ótimas, i.e., soluções de qualidade inferior à qualidade dos ótimos globais. Isso acontece porque a diversidade das soluções produzidas por um algoritmo guloso é muito pequena. Caso não existam elementos com valores idênticos produzidos pela função gulosa, ou caso uma regra determinística seja usada para selecionar entre esses elementos, o algoritmo guloso produzirá sempre a mesma solução. Para garantir diversidade de soluções e ao mesmo tempo, um controle na qualidade das soluções produzidas, a metaheurística GRASP usa um algoritmo não totalmente guloso, pois ele não vai escolher

necessariamente o melhor elemento para entrar na solução, mas sim um elemento entre os melhores, para produzir as soluções iniciais usadas em cada iteração.

A figura 5.3 mostra o pseudo-código de um procedimento GRASP no qual são realizados MAX_ITER iterações.

A seguir, são propostos um algoritmo de construção, ADD, dois algoritmos de busca local, o primeiro que chamamos de BL1 e o segundo, busca local BL2 e mais um algoritmo de Reconexão de Caminhos, mais conhecido na literatura como *Path Relinking* (PR).

5.1.1 Algoritmo de Construção

Usamos um algoritmo de construção por adição de vértices denominado ADD. Baseado em um critério de inserção, de tal modo que a solução parcial comece a ser construída a partir de uma rota inicial contendo todos os vértices de T, gerada pela heurística do Caixeiro Viajante conhecida como Heurística de Inserção Mais Barata.

A partir de LDisp, lista dos vértices não pertencentes à rota parcial, é montada a Lista de Candidatos Restrita (LCR), com os p melhores elementos de LDisp, onde $p = 30\%$ do tamanho de LDisp. Os vértices são inseridos em LCR de acordo com o seguinte critério de seleção:

$$P(k,i,j) = \frac{Cp_k + B\phi(k)}{A[c_{ij} - (c_{ik} + c_{kj})]}$$

para cada $k = 1, ..., n$ e $i, j = 1, ..., n$ e $A, B, C \geq 0$

Onde:

- $P(k,i,j)$ fornece o benefício de inserir o vértice k entre os vértices adjacentes i e j da rota corrente;
- k é o índice do vértice candidato à inserção;
- i, j são os índices dos vértices da rota parcial entre os quais o vértice k será inserido;

- p_k é o prêmio de k;
- c_{ij}, c_{ik}, c_{kj} são as distâncias Euclidianas entre os respectivos vértices;
- A, B, C, constantes inteiras;
- $\phi(k)$ representa o número de vértices do conjunto W que são cobertos por k, e não são cobertos pelos outros vértices da rota parcial.

Os vértices são inseridos na rota parcial até que uma rota viável para o PRR-CP tenha sido obtida, i.e., todo $w \in W$ esteja coberto e o prêmio mínimo tenha sido coletado.

```
1.  Algoritmo ADD: Heurística de Construção por Adição de Vértices
2.  Entrada:
3.    ROTA ← {v | v ∈ T}
4.    CUSTO-ROTA ← CUSTO-ROTA (ROTA);
5.    Pcolet ← Σ_{j∈T} P_j;
6.    LCR ← n melhores elementos de LDisp;
7.     Enquanto (critério de parada não satisfeito) faça
8.      ESCOLHIDO ← vértice escolhido aleatoriamente de LCR;
9.      Pcolet← Pcolet + Prêmio associado à ESCOLHIDO;
10.     atualiza LCR;
11.     POSIÇÃO ← melhor posição para inserir ESCOLHIDO na ROTA;
12.     ROTA ← insere-rota (ROTA, POSIÇÃO);
13.     CUSTO-ROTA ← recalcula CUSTO-ROTA (ROTA);
```

Figura 5.4: Algoritmo para construção de uma solução inicial por adição de vértices

Na linha 3, do algoritmo ADD, figura 5.4, obtemos a rota parcial inicial, formada pelos vértices de T. Ainda que esta rota seja inviável, a inserção dos demais vértices é capaz de ajustá-la. Na linha 4, inicializamos o custo da rota, com o custo da rota parcial formada somente pelos vértices de T. Na linha 6, inicializamos a Lista de Candidatos Restrita com os n melhores vértices disponíveis para adição, LDisp. De LDisp fazem parte todos os vértices do grafo não pertencentes à rota. O algoritmo

prossegue com a escolha de um vértice de LCR até que o critério de parada seja atendido, ou seja, até que todo $w \in W$ esteja coberto e $PRIZE$ tenha sido obtido (linha 7). Na linha 9, atualizamos o total de prêmios coletado até o momento. Na linha 10, atualizamos LCR, uma vez que o número de vértices descobertos muda a cada iteração. Na linha 11, a variável POSIÇÃO armazena o par ordenado entre o qual ESCOLHIDO será inserido, que vai ser determinada por uma heurística do Problema do Caixeiro Viajante (PCV). Neste caso, a heurística de Inserção mais Barata. Na linha 12, a função insere-rota vai inserir o vértice ESCOLHIDO, na ROTA, na POSIÇÃO determinada. Na linha 13, a função recalcula CUSTO-ROTA vai atualizar o custo da rota.

5.1.2 Algoritmos de Busca Local

A busca local, normalmente, é a fase mais dispendiosa, em termos de tempo computacional, de uma metaheurística. É nesta fase que vários ótimos locais são analisados a fim de se tentar obter um ótimo global. Descrevemos a seguir, duas propostas de busca local para o PRR-CP.

Algoritmo de Busca Local - BL1

Dada uma solução viável efetuar todas as trocas do tipo:

- Colocar um $v_1 \in V \backslash T$, mas não pertencente à rota atual e,
- Retirar $v_2 \in V \backslash T$, pertencente à rota atual.

A cada permutação avalia-se a viabilidade da solução resultante. Este algoritmo é executado até que uma solução melhor do que a solução inicial, construída na primeira fase do GRASP, seja encontrada. Note que no pior caso toda a vizinhança da solução inicial terá que ser percorrida.

Na linha 3, do algoritmo BL1, figura 5.5, define-se o conjunto V das vizinhanças, este conjunto é inicializado com todos os vértices que não fazem parte da solução

1. **Algoritmo BL1:**
2. **Entrada**
3. Definir o conjunto de $V = \{v \in V \backslash T$, mas nÃ£o pertencente à rota atual$\}$
4. Rota-Inicial ← solução inicial viável, fornecida por uma heurística de construção;
5. Rota-Atual ← Rota-Inicial;
6. Melhor-Rota ← Rota-Atual;
7. Pcolet $\leftarrow \sum_{j \in ROTA} P_j$,
8. Custo-Atual ← custo associado à Rota -Inicial;
9. Melhor-Custo ← Custo-Atual;
10. **Enquanto** (critério de parada não satisfeito) **faça**
11. Gerar uma solução Rota-Nova na vizinhança V de Rota-Atual;
12. Rota-Atual ← Rota-Nova;
13. Custo-Atual ← custo associado à Rota-Nova;
14. **Se** Custo-Atual < Melhor-Custo **então**
15. Melhor-Rota ← Rota-Atual;
16. Melhor-Custo ← Custo-Atual;

Figura 5.5: Algoritmo de busca local baseado na remoção de vértices seguida de sucessivas adições

fornecida pela heurística de construção, na qual a busca local será realizada. Na linha 4, inicializa-se a variável Rota-Inicial, nela armazena-se a solução fornecida pela busca local. Na linha 5, inicializa-se a variável Rota-Atual com Rota-Inicial. Na linha 6, inicializa-se Melhor-Rota com Rota-Atual. Na linha 7, inicializa-se a variável Pcolet, com o prêmio coletado pela Rota-Inicial. Na linha 8, o Custo-Atual é inicializado com o custo da Rota-Inicial. Na linha 9, o Melhor-Custo é inicializado com o Custo-Atual. O algoritmo prossegue fazendo as trocas das vizinhanças até que uma solução melhor do que a solução inicial seja encontrada, linha 10. Na linha 11, uma Rota-Nova (viável) na vizinhança V da Rota-Atual é gerada da seguinte forma, retira-se um vértice optativo de Rota-Atual e coloca-se um vértice da vizinhança em Rota-Nova. Neste ponto também checamos a viabilidade de Rota-Nova, ou seja, se ela cobre todos os vértices de W e se o prêmio mínimo ($PRIZE$) é coletado. Na linha 12, atualiza-se a Rota-Atual com a Rota-Nova, que acabou de ser gerada. Na

linha 13, atualiza-se o Custo-Atual com o custo da Rota-Nova. Se o custo da Rota-Atual (Custo-Atual) for melhor do que o custo da Melhor-Rota (Melhor-Custo), linha 14, então deve-se atualizar a Melhor-Rota e o Melhor-Custo, linhas 15 e 16, respectivamente.

Algoritmo de Busca Local - BL2

Baseados na metaheurística VNS (*Variable Neighbourhood Search*) proposta por Hansen e Mladenovic [29], que consiste basicamente em uma sistemática troca de vizinhança associada a um algoritmo randômico de busca local, desenvolvemos nosso algoritmo de busca. Contrariamente a outras metaheurísticas baseadas em métodos de busca local, VNS não segue uma trajetória, mas explora incrementalmente vizinhanças distantes da solução corrente, atualizando esta solução somente quando uma melhora puder ser realizada. Neste caso, freqüentes características da solução atual são guardadas e utilizadas para a obtenção de soluções vizinhas promissoras.

Apesar do nosso algoritmo de busca local se assemelhar a um VNS, ele não pode ser chamado de VNS pois não explora incrementalmete as vizinhanças de uma solução inicial. Ao invés disso, ele explora a vizinjança $V1$ da solução e ao encontrar uma melhor solução nesta vizinhança o algoritmo passará a explorar a vizinhança $V2$ desta segunda solução, encontrando uma melhor solução em $V2$, o algoritmo passará a explorar a vizinhança $V3$ desta terceira solução.

Desta forma, as vizinhanças de busca associadas a uma solução são definidas do seguinte modo:

- $V_1 = \{$colocar um vértice não pertencente à solução atual e retirar um vértice da solução atual.$\}$;
- $V_2 = \{$colocar dois vértices não pertencentes à solução atual e retirar dois vértices da solução atual.$\}$;
- $V_n = \{$colocar n vértices não pertencentes à solução atual e retirar n vértices da solução atual.$\}$; onde, o valor de n pode variar até $\mid V \setminus T \mid$.

1. **Algoritmo BL-2:**
2. **Entrada**
3. Definir o conjunto de estruturas de vizinhança V_n
4. Rota-Inicial ← solução inicial viável para o PRR-CP,fornecida por uma heurística de construção;
5. Rota-Atual ← Rota-Inicial;
6. Melhor-Rota ← Rota-Atual;
7. Pcolet ← $\sum_{j \in ROTA} P_j$;
8. Custo-Atual ← custo associado à Rota -Inicial;
9. Melhor-Custo ← Custo-Atual;
10. $n \leftarrow 1$;
11. **Enquanto** ($n \leq n_{max}$) **faça**
12. **Enquanto** (critério de parada não satisfeito) **faça**
13. Gerar uma solução Rota-Nova na vizinhança V_n de Rota-Atual;
14. Rota-Atual ← Rota-Nova;
15. **Se** Custo-Atual < Melhor-Custo **então**
16. Melhor-Rota ← Rota-Atual;
17. Melhor-Custo ← Custo-Atual;
18. $n \leftarrow n + 1$;

Figura 5.6: Algoritmo de busca local BL2

Na linha 3, do algoritmo BL-2, figura 5.6, defini-se o conjunto V_n das vizinhanças, este conjunto é inicializado com todos os vértices que não fazem parte da solução fornecida pela heurística de construção. Na linha 4, inicializa-se a variável Rota-Inicial, nela armazena-se a solução fornecida pela busca local. Na linha 5, inicializa-se a variável Rota-Atual com Rota-Inicial. Na linha 6, inicializa-se Melhor-Rota com Rota-Atual. Na linha 7, inicializa-se a variável Pcolet com o prêmio coletado pela Rota-Inicial. Na linha 8, o Custo-Atual é inicializado com o custo da Rota-Inicial. Na linha 9, o Melhor-Custo é inicializado com o Custo-Atual. O *loop* mais externo, linha 11, será executado tantas vezes quantas forem o número de estruturas de vizinhanças de BL-2. Por isso inicializamos a variável inteira n com 1, na linha 10 e a incrementamos na linha 19. O algoritmo prossegue fazendo as trocas das vizinhanças até todo o conjunto V_n de vizinhos tenha sido percorrido, linha 12.

Na linha 13, uma Rota-Nova (viável) na vizinhança V_n da Rota-Atual é gerada da seguinte forma, retiram-se n vértices optativos de Rota-Atual e colocam-se n vértices da vizinhança em Rota-Nova. Neste ponto também checamos a viabilidade de Rota-Nova, ou seja, se ela cobre todos os vértices de W e se o prêmio mínimo ($PRIZE$) é coletado. Na linha 14, atualiza-se a Rota-Atual com a Rota-Nova, que acabou de ser gerada. Na linha 15, atualiza-se o Custo-Atual com o custo da Rota-Nova. Se o custo da Rota-Atual (Custo-Atual) for melhor do que o custo da Melhor-Rota (Melhor-Custo), linha 16, então deve-se atualizar a Melhor-Rota e o Melhor-Custo, linhas 17 e 18, respectivamente.

5.1.3 Reconexão de caminhos

A Reconexão de Caminhos (RC) foi inicialmente introduzida no contexto da metaheurística Busca Tabu [26], como uma abordagem para integrar estratégias de intensificação e diversificação ao processo de busca. Em [25] é apresentado um levantamento sobre reconexão de caminhos. A técnica consiste em explorar trajetórias que conectam soluções de alta qualidade, começando de uma *solução base* e gerando um caminho na vizinhança dessa solução na direção de outra solução, chamada de *solução alvo*. Esse caminho é gerado selecionando-se movimentos que introduzam atributos da solução alvo na solução base. A cada passo, todos os movimentos que incorporam atributos da solução alvo são analisados e o melhor movimento é escolhido. Essa abordagem é denominada reconexão de caminhos porque quaisquer duas soluções visitadas por um algoritmo de busca tabu estão ligadas por uma seqüência de movimentos.

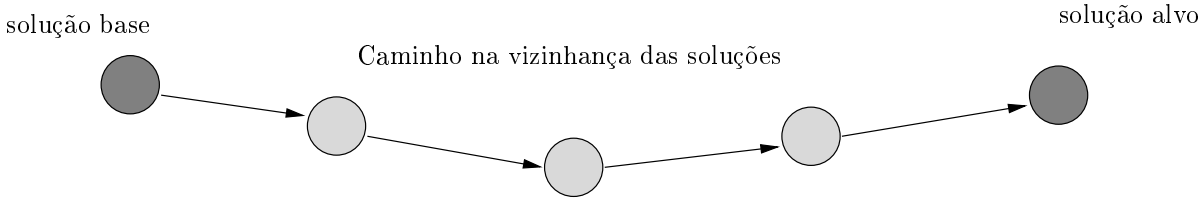

Figura 5.7: Reconexão de caminhos: exploração de trajetórias que conectam soluções de alta qualidade (elite)

A reconexão de caminhos incorporada a um algoritmo GRASP tem como objetivo intensificar a busca em regiões onde soluções de qualidade tenham sido encontradas. Seu uso foi introduzido por Laguna e Martí [33]. Na implementação proposta por eles, as três melhores soluções obtidas pela estratégia são armazenadas em um conjunto elite P para serem usadas como soluções alvo pela reconexão de caminhos. Após cada iteração do algoritmo GRASP, a solução obtida é comparada às soluções armazenadas em P. Caso a nova solução seja melhor do que alguma das soluções armazenadas, o conjunto elite P é atualizado.

A RC pode ser incorporada a uma abordagem GRASP de duas formas básicas. A primeira delas consiste em aplicar a reconexão após cada iteração do algoritmo, entre a solução produzida pela busca local e uma solução do conjunto elite corrente. A segunda estratégia consiste em realizar a reconexão entre soluções do próprio conjunto elite, obtidas durante as iterações do GRASP. A figura 5.8 mostra o pseudo-código de um procedimento GRASP com reconexão de caminhos.

```
1.  procedimento GRASP COM RECONEXÃO DE CAMINHO
2.    Repita por Max_Iterations
3.      Construa uma solução randômica gulosa;
4.      Use busca local para tentar melhorar a qualidade da solução construída;
5.      Aplique reconexão de caminhos para melhorar a qualidade da solução;
6.      Atualize o conjunto elite;
7.      Atualize a melhor solução encontrada;
```

Figura 5.8: Algoritmo básico de GRASP com reconexão de caminho.

Neste trabalho, a Reconexão de Caminhos (RC) é feita usando-se a solução produzida pela busca local como solução base e todas as soluções de P, escolhidas como alvo. Por este motivo, a fim de não sobrecarregarmos o GRASP com a adição deste módulo de RC, utilizamos um conjunto elite bem pequeno, neste caso de tamanho 3. Além disso, ainda pelo mesmo motivo de não sobrecarregar o tempo de processamento do algoritmo, não vamos ativar a RC em todas as iterações do GRASP. Optamos por ativar a RC de tempos em tempos, apenas quando conjunto

elite P tiver sido atualizado.

Para fazer a reconexão nós separamos os vértices de $V \backslash T$ da solução base e da solução alvo. A cada iteração da reconexão, um vértice de $V \backslash T$ pertencente à solução alvo é introduzido na solução base. Note que neste passo nenhum processamento muito grande é feito na escolha do vértice que vai entrar na solução base, apenas pegamos o primeiro vértice sa solução alvo e o acrescentamos na solução base. Fazemos isso com o intuito de não onerarmos o tempo de processamento do algoritmo como um todo. Após se introduzir um vértice na solução base, o algoritmo irá tentar retirar os vértices de $V \backslash T$ não pertencentes à solução alvo e irá avaliar o custo dessa solução gerada. Somente a melhor solução gerada é guardada. Note que, a remoção destes vértices só será possível caso esse passo não gere uma solução inviável, ou seja, caso nenhum vértice de W fique descoberto e PRIZE seja coletado. Portanto, em nenhum passo da reconexão uma solução inviável é gerada.

Capítulo 6

Resultados Computacionais

Nesta seção apresentamos os resultados computacionais com os dois procedimentos GRASP propostos. O GRASP que chamamos de GRASP1, consiste do algoritmo de construção ADD e da busca local BL1. E o GRASP2, consiste do algoritmo de construção ADD e da busca local BL-VNS.

Uma comparação entre os resultados obtidos pelas heurísticas e o ótimo fornecido pelo método exato, usando a formulação matemática proposta, e ainda os resultados obtidos com a aplicação das regras de redução nas instâncias testadas no método exato são apresentados. Nos nomes das instâncias, os prefixos numéricos indicam o tamanho da mesma.

Devido à ausência de instâncias para o PRR ou o PRR-CP nas bibliotecas der instâncias na Internet. Foi implementado um conjunto de instâncias para testes usando um plano cartesiano de 100 x 100, as coordenadas foram geradas aleatórimanente, e a distância usada é a euclidiana.

Os testes foram realizados em um micro computador Atlhon, com processador de 2GHz e 1Gb de memória RAM.

As tabelas 6.1, 6.2, 6.3 e 6.4 mostram a taxa de redução obtida com a aplicação das regras aqui propostas. Nas segunda, terceira e quarta colunas temos as dimensões de cada conjunto associado ao PRR-CP, $|W|$, conjunto dos vértices que

devem ser cobertos, $|V \setminus T|$, conjunto dos vértices que podem entrar na rota e $|T|$, conjunto dos vértices que devem estar na rota. Na quinta coluna temos o número total de vértices do Grafo Original (GO). Na sexta coluna temos o tamanho do Grafo Reduzido (GR), que consiste no grafo original após a aplicação das regras de redução. E finalmente, na última coluna temos o percentual de redução dos grafos, após o uso das regras propostas.

Os resultados das tabelas 6.1 - 6.4 mostram o impacto das regras de redução aqui propostas para o PRR-CP, onde em média houve uma redução de 20% para grafos de diferentes dimensões.

6.1 Avaliação das heurísticas GRASP

Esta seção apresenta os resultados computacionais obtidos com as heurísticas GRASP propostas para instâncias de até 500 vértices. As tabelas 6.5 e 6.6 mostram o desvio das metaheurísticas para os grafos originais e reduzidos. Este desvio é calculado entre a média das soluções obtidas pelo algoritmo (executados três vezes cada) e a melhor média considerando todos as heurísticas:

$$[\ (\ \textit{média das soluções - melhor solução heurística}) \div \textit{média das soluções}]\ .$$

Em 6.5, da segunda à quinta coluna, apresentamos os resultados obtidos pelo GRASP1, com os grafos originais (GO) e reduzidos (GR), com e sem a aplicação da Reconexão de Caminhos (RC), respectivamente. Em 6.6 mostramos os mesmos resultados para o GRASP2. Note que, em muitos casos a qualidade da solução melhora para os grafos reduzidos (GR). Isso acontece devido à diminuição no tamanho do grafo de entrada e consequentemente na diminuição do espaço de busca das soluções. Facilitando, dessa forma, o trabalho da heurística para encontrar soluções de boa qualidade.

As tabelas 6.7 e 6.8 mostram o tempo de processamento das heurísticas GRASP1 e GRASP2, para os grafos originais e reduzidos, com e sem a aplicação da Reconexão de Caminhos (RC), respectivamente. Neste caso, em particular, com o objetivo de

TESTE	$\|W\|$	$\|V \setminus T\|$	$\|T\|$	$\|W \bigcup V\|$ -GO	$\|W \bigcup V\|$ -GR	% redução
1	9	6	15	30	24	20
2	15	9	6	30	25	17
3	6	12	12	30	22	27
4	3	15	12	30	21	30
5	12	12	6	30	20	33
6	3	12	15	30	18	40
7	15	12	3	30	20	33
8	12	9	9	30	20	33
9	3	9	18	30	18	40
10	6	9	15	30	18	40
11	15	10	25	50	43	14
12	25	15	10	50	45	10
13	10	20	20	50	42	16
14	5	25	20	50	41	18
15	20	20	10	50	44	12
16	5	20	25	50	41	18
17	25	20	5	50	45	10
18	25	15	10	50	45	10
19	20	15	15	50	44	12
20	10	15	25	50	42	16
21	20	50	30	100	100	0
22	50	30	20	100	85	15
23	20	40	40	100	82	18
24	10	50	40	100	78	22
25	40	40	20	100	84	16
26	10	40	50	100	78	22
27	50	40	10	100	85	15
28	40	30	30	100	84	16
29	10	30	60	100	78	22
30	20	30	50	100	82	18

Tabela 6.1: Redução do Grafo, com até 100 vértices, usando as regras propostas para o PRR-CP

TESTE	$\|W\|$	$\|V \setminus T\|$	$\|T\|$	$\|W \bigcup V\|$ -GO	$\|W \bigcup V\|$ -GR	% redução
31	45	30	75	150	124	17
32	75	45	30	150	125	17
33	30	60	60	150	122	18
34	15	75	60	150	118	21
35	60	60	30	150	124	17
36	15	60	75	150	118	21
37	75	60	15	150	125	17
38	60	45	45	150	124	17
39	15	45	90	150	118	21
40	30	45	75	150	122	18
41	40	100	60	200	200	0
42	100	60	40	200	170	15
43	60	100	40	200	200	0
44	20	100	80	200	149	26
45	80	80	40	200	169	16
46	20	80	100	200	163	19
47	100	80	20	200	170	15
48	80	60	60	200	169	16
49	20	60	120	200	113	44
50	40	60	100	200	117	42

Tabela 6.2: Redução do Grafo, com até 200 vértices, usando as regras propostas para o PRR-CP

TESTE	$\|W\|$	$\|V \setminus T\|$	$\|T\|$	$\|W \bigcup V\|$ -GO	$\|W \bigcup V\|$ -GR	% redução
51	75	50	125	250	197	21
52	125	75	50	250	175	25
53	50	100	100	250	170	32
54	25	125	100	250	163	35
55	100	100	50	250	199	20
56	25	100	125	250	193	23
57	125	100	25	250	200	20
58	100	75	75	250	200	20
59	25	75	150	250	193	23
60	50	75	125	250	197	21
61	90	60	150	300	236	21
62	90	150	60	300	300	0
63	60	120	120	300	220	27
64	30	150	120	300	212	29
65	120	120	60	300	200	33
66	30	120	150	300	195	35
67	150	120	30	300	200	33
68	120	90	90	300	209	30
69	30	90	180	300	204	32
60	60	90	150	300	207	31

Tabela 6.3: Redução do Grafo, com até 300 vértices, usando as regras propostas para o PRR-CP

TESTE	$\|W\|$	$\|V \setminus T\|$	$\|T\|$	$\|W \bigcup V\|$ -GO	$\|W \bigcup V\|$ -GR	% redução
71	120	80	200	400	316	21
72	200	120	80	400	301	25
73	80	160	160	400	295	26
74	120	200	80	400	400	0
75	160	160	80	400	250	38
76	40	160	200	400	245	39
77	200	160	40	400	250	38
78	160	120	120	400	279	30
79	40	120	240	400	274	32
80	80	120	200	400	277	31
81	150	100	250	500	396	21
82	250	150	100	500	374	25
83	150	250	100	500	500	0
84	50	250	200	500	362	28
85	200	200	100	500	350	30
86	50	200	250	500	344	31
87	250	200	50	500	500	0
88	200	150	150	500	350	30
89	50	150	300	500	344	31
90	100	150	250	500	348	30
91	300	200	500	1000	796	20
92	500	300	200	1000	749	25
93	200	400	400	1000	745	26
94	100	500	400	1000	737	26
95	400	400	200	1000	650	35
96	100	400	500	1000	645	36
97	500	400	100	1000	650	35
98	400	300	300	1000	699	30
99	100	300	600	1000	694	31
100	200	300	500	1000	697	30

Tabela 6.4: Redução do Grafo, com até 1.000 vértices, usando as regras propostas para o PRR-CP

TESTE	GRASP1(GO)	GRASP1(GR)	GRASP1+RC(GO)	GRASP1+RC(GR)
50.a	0,0000	0,0000	0,0000	0,0000
50.b	0,0000	0,0000	0,0000	0,0000
50.c	0,0000	0,0000	0,0000	0,0000
100.a	0,0014	0,0014	0,0000	0,0014
100.b	0,0028	0,0028	0,0042	0,0028
100.c	0,0014	0,0014	0,0014	0,0014
200.a	0,0243	0,0321	0,0038	0,0107
200.b	0,0000	0,0000	0,0000	0,0000
200.c	0,0230	0,0160	0,0010	0,0000
300.a	0,0817	0,0933	0,0156	0,0082
300.b	0,0000	0,0000	0,0000	0,0000
300.c	0,0000	0,0000	0,0000	0,0000
400.a	0,0000	0,0000	0,0000	0,0000
400.b	0,0351	0,0351	0,0056	0,0042
400.c	0,1115	0,0921	0,0180	0,0083
500.a	0,0469	0,0515	0,0017	0,0034
500.b	0,0640	0,0586	0,0035	0,0029
500.c	0,0411	0,0512	0,0000	0,0012
Média	0,0206	0,02419	0,0030	0,0024

Tabela 6.5: Desvio obtido pela heurística GRASP1 em instâncias de médio e grande porte

não sobrecarregarmos o tempo de processamento das heurísticas foi implementada um RC suavizada. Ou seja, o tamanho do conjunto elite é bem pequeno e esta não é aplicada em todas as iterações do GRASP. Além disso, não foi pesquisado a cada passo da RC o melhor movimento a ser feito. Ao invés disso, pegamos o primeiro vértice de $V \backslash T$ da lista de vértices que pertencem à solução alvo, mas não pertence à solução base e insere-se este vértice na solução base. Feito isso, tentamos retirar os demais vértices que pertencem a solução base, mas não pertencem a solução alvo.

TESTE	GRASP2(GO)	GRASP2(GR)	GRASP2+RC(GO)	GRASP2+RC(GR)
50.a	0,0000	0,0000	0,0000	0,0000
50.b	0,0000	0,0000	0,0000	0,0000
50.c	0,0000	0,0000	0,0000	0,0000
100.a	0,0014	0,0014	0,0000	0,0014
100.b	0,0056	0,0028	0,0000	0,0028
100.c	0,0000	0,0014	0,0014	0,0014
200.a	0,0321	0,0321	0,0000	0,0077
200.b	0,0000	0,0000	0,0000	0,0000
200.c	0,0160	0,0200	0,0030	0,0040
300.a	0,0858	0,0809	0,0148	0,0000
300.b	0,0000	0,0000	0,0000	0,0000
300.c	0,0000	0,0000	0,0000	0,0000
400.a	0,0000	0,0000	0,0000	0,0000
400.b	0,0358	0,0386	0,0084	0,0000
400.c	0,1018	0,1060	0,0450	0,0000
500.a	0,0538	0,0504	0,0000	0,0074
500.b	0,0634	0,0598	0,0000	0,0035
500.c	0,0443	0,0449	0,0031	0,0006
Média	0,0244	0,02435	0,0042	0,0016

Tabela 6.6: Desvio obtido pela heurística GRASP2 em instâncias de médio e grande porte

TESTE	GRASP1(GO)	GRASP1(GR)	GRASP1+RC(GO)	GRASP1+RC(GR)
50.a	0,0000	0,0000	0,0100	0,0125
50.b	0,0000	0,0000	0,1900	0,1875
50.c	0,0000	0,0000	0,2350	0,2450
100.a	1,0000	1,0000	1,5500	1,5600
100.b	0,0000	0,0000	0,5975	0,6225
100.c	1,0000	1,0000	1,4000	1,3900
200.a	13,0000	13,0000	13,7400	13,7600
200.b	0,0000	0,0000	0,1200	0,1200
200.c	13,8050	13,5500	13,8300	13,5100
300.a	30,0000	29,0000	30,0000	30,1200
300.b	0,0000	0,0000	0,3300	0,3200
300.c	41,5000	41,0000	41,8300	41,6900
400.a	63,5000	0,0000	60,3600	0,4000
400.b	26,0000	26,0000	26,6500	26,3000
400.c	75,7500	74,5000	76,1800	75,1900
500.a	126,0000	127,5000	130,0000	125,0400
500.b	248,2500	247,7500	246,2900	248,7700
500.c	108,7500	109,5000	109,7200	109,3300

Tabela 6.7: Tempo médio de processamento,em segundos, para GRASP1

TESTE	GRASP2(GO)	GRASP2(GR)	GRASP2+RC(GO)	GRASP2+RC(GR)
50.a	0,0100	0,0100	0,0125	0,0125
50.b	0,1900	0,1925	0,1900	0,1950
50.c	0,2375	0,2425	0,2475	0,2400
100.a	1,5300	1,5700	1,5700	1,5600
100.b	0,5875	0,5600	0,5600	0,6250
100.c	1,3900	1,3900	1,3900	1,5300
200.a	13,7200	13,8500	13,7300	13,8400
200.b	0,1100	0,1000	0,1200	0,1200
200.c	13,0000	13,0000	13,7600	13,5900
300.a	30,0400	29,9300	30,0200	28,6900
300.b	0,3000	0,2900	0,3300	0,3100
300.c	41,8900	41,8100	41,9800	41,8400
400.a	63,1100	0,3800	60,3600	0,4000
400.b	26,6500	26,2700	26,3700	26,3400
400.c	76,2900	75,0000	75,9900	75,0000
500.a	126,4800	126,4700	126,7000	126,3700
500.b	248,2000	247,2500	247,6900	249,8600
500.c	114,7800	115,3500	109,4000	109,4000

Tabela 6.8: Tempo médio de processamento,em segundos, para GRASP2

TESTE	GRASP1(GO)	GRASP1(GR)	GRASP1+RC(GO)	GRASP1+RC(GR)
10.a	0	0	0	0
10.b	0	0	0	0
10.c	0	0	0	0
15.a	0	0	0	0
15.b	0	0	0	0
15.c	0	0	0	0
20.a	0	0	0	0
20.b	0	0	0	0
20.c	0	0	0	0
30.a	0	0	0	0
30.b	0	0	0	0
30.c	0,069	0,069	0	0
40.a	0	0	0	0
40.b	0	0	0	0
40.c	0,070	0,070	0	0
Média	0,0092	0,0092	0	0

Tabela 6.9: Desvio entre as soluções obtidas pela heurística GRASP1 e as soluções exatas

6.2 Comparação entre o GRASP e o método exato

Nesta seção apresentamos uma comparação entre as heurísticas aqui propostas e o método exato, utilizando a formulação matemática proposta. Nas tabelas 6.9 e 6.10, a primeira coluna indica o tamanho da instância testada, a segunda e terceira coluna indica o desvio das soluções heurísticas em relação ao ótimo global, fornecido pelo XPRESS-MP [15]. Note nas tabelas 6.9 e 6.10 que o GRASP sem RC conseguiu chegar no ótimo para quase todas as instâncias testadas. Enquanto que o GRASP com RC conseguiu atingir o ótimo para todas as instâncias. Nestes exemplos, não pode ser verificado um impacto significativo nos grafos reduzidos (GR) devido ao tamanho das instâncias, que são bem pequenas.

TESTE	GRASP2(GO)	GRASP2(GR)	GRASP2+RC(GO)	GRASP2+RC(GR)
10.a	0	0	0	0
10.b	0	0	0	0
10.c	0	0	0	0
15.a	0	0	0	0
15.b	0	0	0	0
15.c	0	0	0	0
20.a	0	0	0	0
20.b	0	0	0	0
20.c	0	0	0	0
30.a	0	0	0	0
30.b	0	0	0	0
30.c	0,069	0,069	0	0
40.a	0	0	0	0
40.b	0	0	0	0
40.c	0,070	0,070	0	0
Média	0,0092	0,0092	0	0

Tabela 6.10: Desvio entre as soluções obtidas pela heurística GRASP2 e as soluções exatas

6.3 Uso das regras de redução no método exato

Nesta seção apresentamos o impacto das regras de redução no desempenho do método exato que utiliza a formulação aqui proposta e o software XPRESS-MP [15]. Na tabela 6.11, na primeira coluna temos o tamanho das instâncias testadas, na segunda coluna temos a performance do método exato sem o uso das regras de redução, e na terceira coluna temos a performance, com o uso das regras. A performance (tempo de execução) é medida em segundos. O (∗) nas colunas indicam que o XPRESS-MP não chegou no ótimo, após quatro horas de processamento.

Na tabela 6.11 podemos observar o impacto das regras de redução no método exato. Para a instância 30.c, por exemplo, com o grafo original não foi possível chegarmos no ótimo. Entretanto com o grafo reduzindo em 218 segundos se obteve a solução ótima. Por outro lado, para a instancia 30.b não houve diminuição alguma no tempo de processamento. Isso ocorreu, pois não houve redução para esta instância.

6.4 Análise do Impacto das regras de redução nas heurísticas e no método exato

Os resultados ilustrados nas tabelas 6.5- 6.11 mostram o impacto do uso das regras de redução propostas na qualidade das soluções geradas pelas heurísticas, bem como nos tempos computacionais exigidos tanto pelas heurísticas quanto pelo método exato.

Além disso, observamos que o uso das regras permite a obtenção de soluções exatas para instâncias de dimensões maiores (tabela 6.11).

Em relação às heurísticas GRASP aqui propostas, verifica-se que tanto o GRASP1 como o GRASP2 apresentam um desempenho muito bom como observado nas comparações com o método exato (tabelas 6.9 e 6.10).

Observamos também que na média a busca local VNS apresenta resultados ligeiramente superiores aos obtidos pela busca BL1 (GRASP1). Outro aspecto relevante,

TESTE	EXATO-GO	EXATO-GR
10.a	0	0
10.b	0	0
10.c	0	0
15.a	0	0
15.b	0	0
15.c	0	0
20.a	6	0
20.b	6	0
20.c	792	613
30.a	3	0
30.b	10.757	10.757
30.c	*	218
40.a	10.756	10.744
40.b	*	15
40.c	3967	219

Tabela 6.11: Performance do método exato com e sem o uso das regras de redução

é a melhora que o módulo de reconexão de caminhos (RC) proporciona ao GRASP, sem onerar significativamente os tempos computacionais.

Em relação aos tempos computacionais do método exato comparando com os tempos das heurísticas observou-se uma diferença significativa onde o XPRESS chegou a exigir em alguns casos mais de 4 horas, o GRASP para estas instâncias exigiu apenas alguns segundos.

Adicionalmente, uma nova bateria de testes foi executada, agora com o intuito de avaliar a robustez das heurísticas aqui propostas em relação a sua convergência para valores alvos sub-ótimos.

6.5 Distribuição de probabilidade das Heurísticas GRASP

O objetivo desta seção é estudar o comportamento médio das heurísticas GRASP e de algumas de suas variantes com reconexão de caminhos. Referências deste estudo podem ser encontradas em [2]. Nesta nova bateria de testes, usamos como critério de parada um valor alvo τ da função objetivo para o GRASP, este vai ser executado até que encontre uma solução com valor pelo menos tão bom quanto τ. Este critério de parada permite comparar os tempos de execução média de cada algoritmo, considerando-se que as soluções obtidas não precisam ser necessariamente idênticas, mas sim de mesma qualidade. O objetivo destes testes é verificar a performance média das heurísticas aqui propostas em termos do tempo necessário para se atingir soluções sub-ótimas.

Para cada par problema teste/valor alvo, mediu-se o tempo de execução necessário para se obter uma solução com custo pelo menos tão bom quanto o valor alvo e analisou-se a distribuição desses tempos.

6.5.1 Projeto dos experimentos

Nessa seção serão descritos os experimentos realizados. Para cada um dos algoritmos GRASP apresentados neste trabalho, a distribuição da probabilidade do tempo para valor alvo foi estudada para três problemas testes. Tomamos como valor alvo a média entre o valor médio obtido pelas heurísticas e o melhor valor da função objetivo. Foram realizados 100 execuções do programa para cada par problema teste/valor alvo. Em cada execução, o programa foi interrompido após encontrar uma solução de custo pelo menos tão bom quanto o valor alvo, e o seu tempo total de CPU (em segundos) foi registrado. Para cada uma das 100 execuções de cada par, o gerador de números aleatórios foi inicializado com uma semente diferente, assim, as execuções são independentes.

Para cada par problema teste/valor alvo, os tempos de execução medidos são ordenados de forma crescente. Ao i-ésimo tempo de execução ordenado (t_i) é associada uma probabilidade $p_i = (i - 1/2)/n$. Os pontos $z_i = (t_i, p_i)$ para $i = 1, ..., n$ são plotados em um gráfico, segundo a definição dada em [2]. Nestes gráficos, cada curva representa o desempenho de um algoritmo e quanto mais a esquerda estiver a curva melhor será desempenho do algoritmo associado. As figuras 6.1 - 6.6 ilustram, separadamente, os gráficos das distribuições exponenciais dos algoritmos GRASP1, GRASP1(Red), GRASP1+PR, GRASP1+PR(Red), GRASP2, GRASP2(Red), GRASP2+PR, GRASP2+PR(Red), para instâncias de tamanho 200, 300 e 400.

O objetivo desta seção é analisar o impacto das regras de redução e do módulo de reconexão de caminhos (RC) nas heurísticas GRASP. Para isso, selecionamos uma instância de 200, 300 e 400 vértices do PRR-CP e para cada instância determinamos um valor alvo a ser atingido.

Pelos resultados observados nas figuras 6.1 a 6.6 notamos que o módulo de reconexão melhora sensivelmente o desempenho do GRASP na sua convergência para valores subótimos.

Por exemplo, na figura 6.2, na avaliação do GRASP2, observamos que as versões

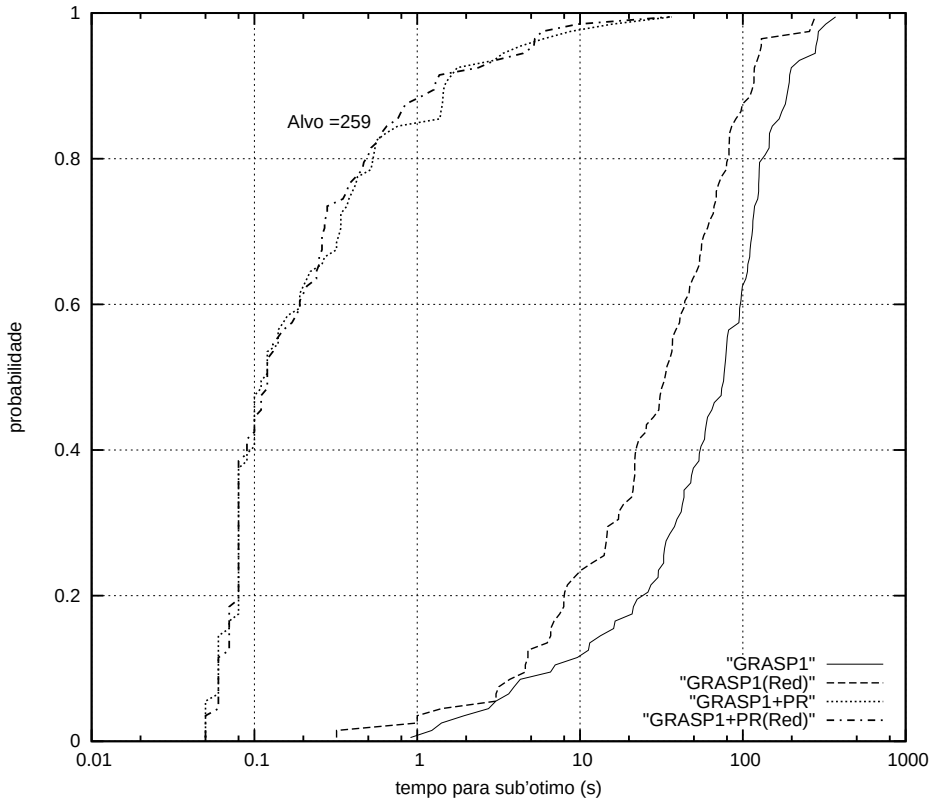

Figura 6.1: Gráfico das distribuições exponenciais para uma instância de tamanho 200, utilizando o algoritmo GRASP1

com RC usando o grafo original (GRASP2+RC) e o grafo reduzido (GRASP2+RC(Red)) tem uma probabilidade de obter uma convergência de 100% em torno de 10 segundos. Neste tempo, as versões sem RC apresentam uma probabilidade de convergência de apenas 10%. Nos demais testes (figuras 6.1, 6.3, 6.4, 6.5 e 6.6) observamos um comportamento similar mostrando com isso, a eficiência do módulo de reconexão de caminhos.

Além disso, os resultados das figuras 6.1 a 6.6 nos mostram também que todas as versões do GRASP, mesmo as mais lentas (GRASP1 e 2 sem RC) convergem para o valor alvo num tempo razoável (menos de 1000 segundos). Isso mostra a robustez destas heurísticas em diferentes testes computacionais, o que dá a elas a confiabilidade necessária para sua utilização na solução de problemas de otimização combinatória.

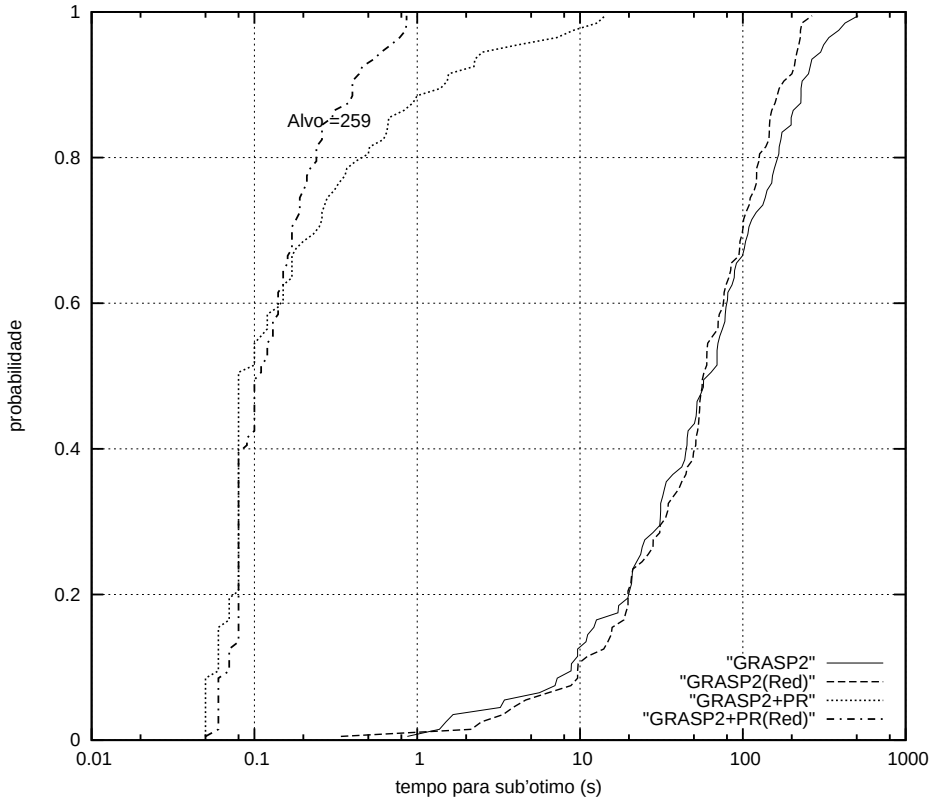

Figura 6.2: Gráfico das distribuições exponenciais para uma instância de tamanho 200, utilizando o algoritmo GRASP2

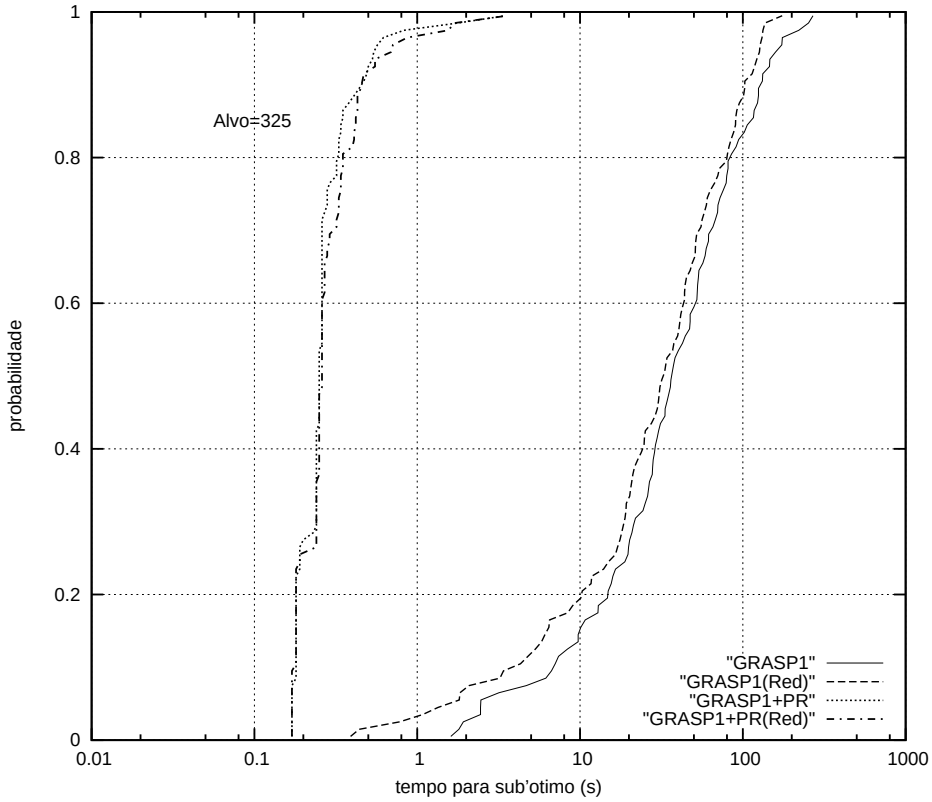

Figura 6.3: Gráfico das distribuições exponenciais para uma instância de tamanho 300, utilizando o algoritmo GRASP1

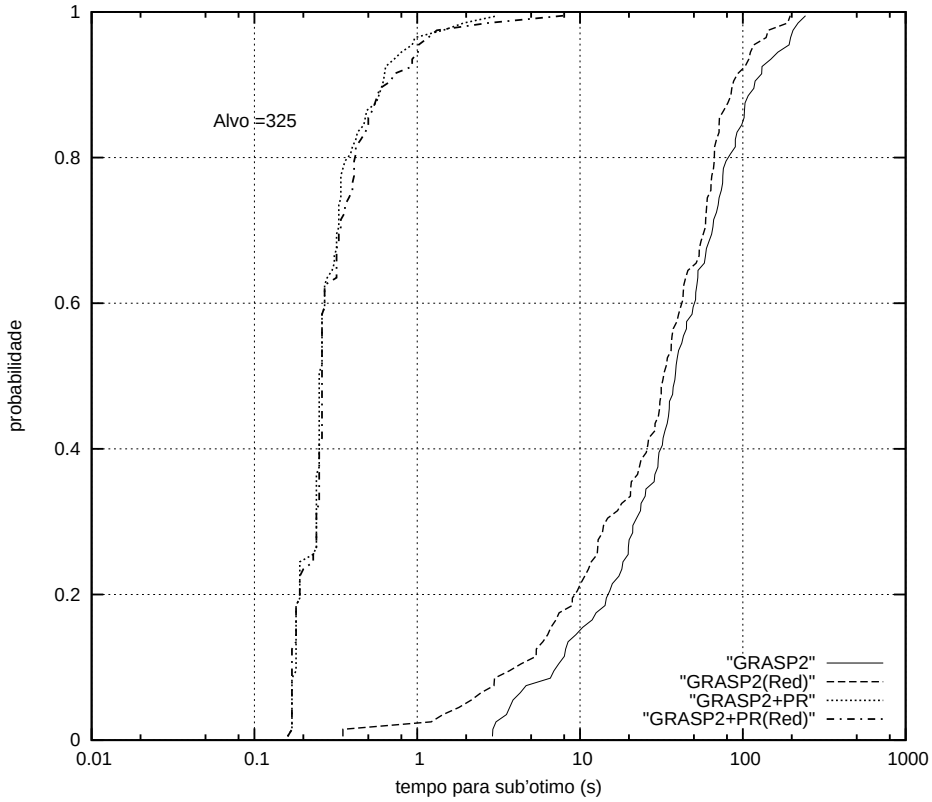

Figura 6.4: Gráfico das distribuições exponenciais para uma instância de tamanho 300, utilizando o algoritmo GRASP2

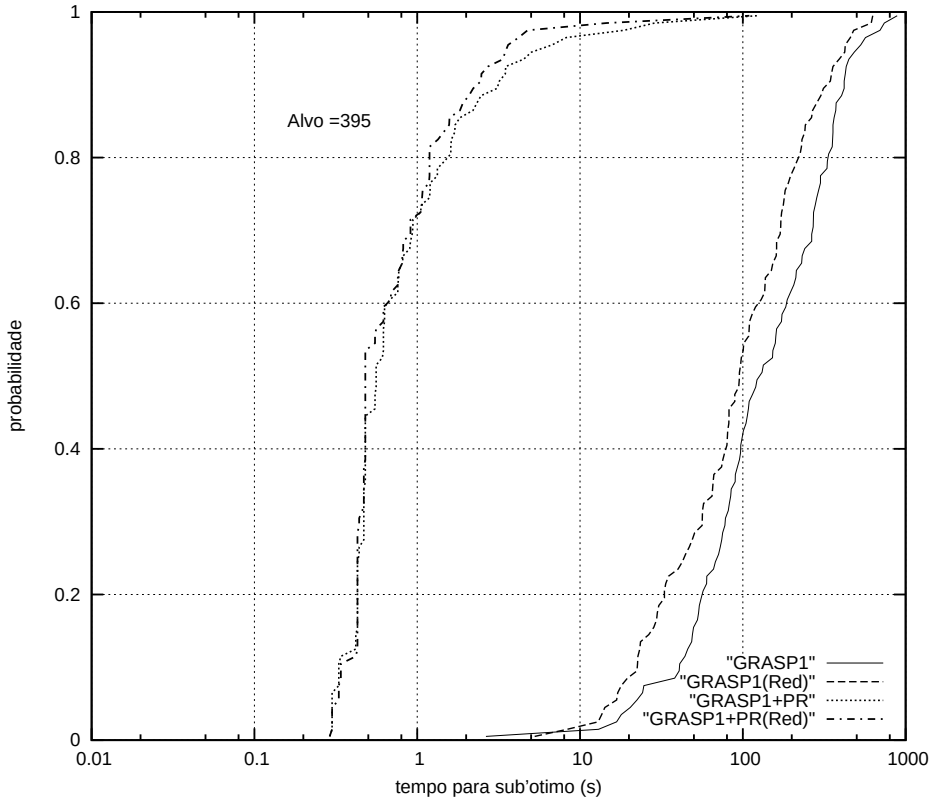

Figura 6.5: Gráfico das distribuições exponenciais para uma instância de tamanho 400, utilizando o algoritmo GRASP1

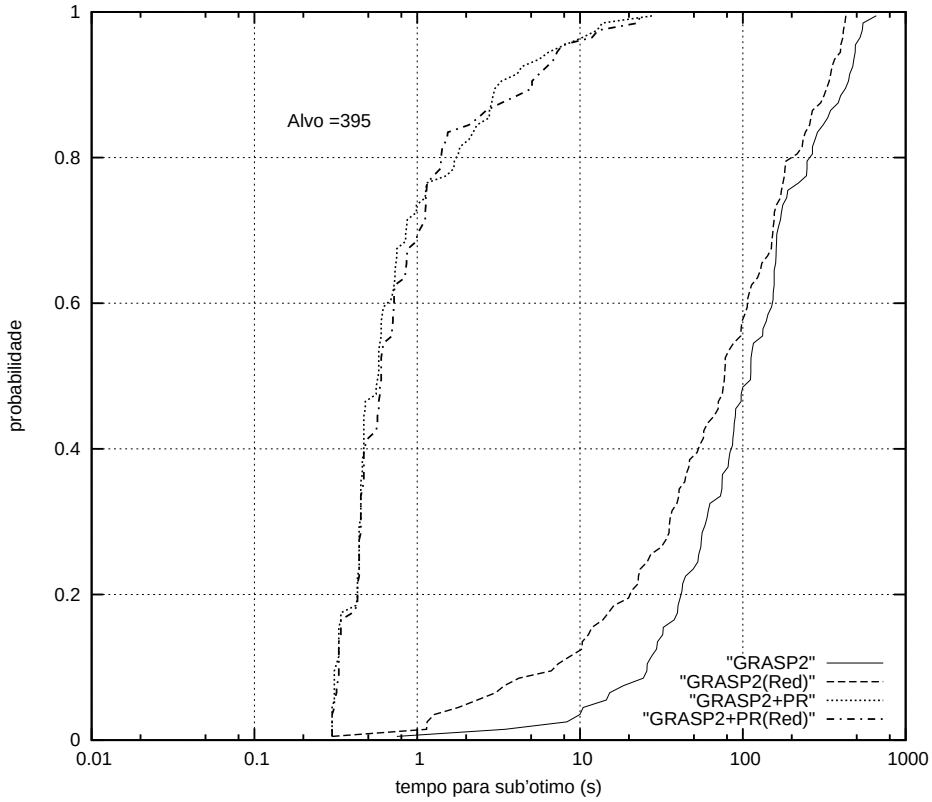

Figura 6.6: Gráfico das distribuições exponenciais para uma instância de tamanho 400, utilizando o algoritmo GRASP2

Capítulo 7

Conclusões e Trabalhos Futuros

Apresenta-se neste trabalho uma nova generalização do TSP definido como "O Problema de Recobrimento de Rotas com Coleta de Prêmios" (PRR-CP). Como contribuições, apresenta-se:

(i) Uma formulação matemática descrevendo-o como um problema de programação linear inteira. A formulação ao contrário da maioria das formulações do TSP exige apenas um número polinomial de restrições para evitar a formação de rotas desconexas da origem;

(ii) São propostas regras para reduzir as dimensões do grafo associado. Nos testes realizados, mostrou-se a importância do uso destas regras que conseguem reduzir em média as dimensões do problema em cerca de 20%;

(iii) Devido à elevada complexidade computacional do problema tratado, são apresentadas heurísticas usando conceitos de GRASP, VNS e Reconexão de Caminhos para resolver aproximadamente o PRR-CP. O desempenho das heurísticas mostrou-se bastante satisfatório encontrando a solução ótima na maioria dos casos onde o ótimo é conhecido. Além disso, mostrou-se a eficácia das regras de redução que melhoram significativamente a performance tanto de métodos exatos como das heurísticas;

(iv) Finalmente, tratamos de uma questão crucial da maioria das heurísticas, que

é seu desempenho irregular para um conjunto de instâncias arbitrárias. Para tratar desta questão mostrou-se através de exaustivas simulações de um grupo de instâncias, o comportamento das heurísticas aqui propostas em relação a sua convergência para valores alvos pré- estabelecidos. Os resultados mostram uma robustez da heurística onde a convergência para 100% das execuções é obtida sempre em tempos computacionais satisfatórios.

Como sugestões para trabalhos futuros, podemos incluir o uso de outras heurísticas para o PRR-CP, o desenvolvimento de outras regras de redução e a paralelização dos algoritmos aqui propostos.

Referências Bibliográficas

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

[2] R. M. Aiex. *Uma investigação experimental da distribuição de probabilidade de tempo de solução de solução em heurísticas GRASP e sua aplicação na análise de implementação paralelas.* PhD thesis, Departamento de Ciência da Computação, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 2002.

[3] R. M. Aiex, M. G. R. Resende,*Parallel Stratagies for GRASP with Path-Relinking*, AT&T Labs Research Technical Report TD-5SQKM9, (2003)

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

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

[6] E. Balas, *The prize collecting traveling salesman problem*, Networks 19 (1989), 621-636.

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

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

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

[10] 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.

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

[12] 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.

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

[14] F. Dalboni, L. Ochi, e L. Drummond. *On improving Evolutionary Algorithms by using Data Mining for the Oil Collector Vehicle Routing Problem*. Proc. of the Int. Network Optimization Conference (INOC 2003), 182-188, France, 2003.

[15] Dash, Biswoth and Northants, *XPRESS-MP*, NN7 3BX, UK, (2003).

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

[17] G. Finke, A. Claus, and E. Gunn, *A two-commodity network flow approach to the traveling salesman problem*, Congress. Numerantium 41, 167-178, (1984).

[18] 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).

[19] 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.

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

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

[22] 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).

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

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

[25] F. Glover, M. Laguna, R. Martí. *Fundamentals of scatter search and path relinking*. Technical report, Graduate School of Business and Administration, University of Colorado, Boulder, CO 80309-0419, 2000.

[26] F. Glover, M. Laguna. *Tabu Seach*. Kluwer Academic Publishers, 1997.

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

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

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

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

[31] P. Kourtz, *The Need for Improved Forest Fire Detection*, Forestry Chronicle, 272-277, (1987).

[32] 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.

[33] M. Laguna, R. Martí. *GRASP and path relinking for 2-layer straight line crossing minimization.* INFORMS Journal on Computing, 11:44-52, 1999.

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

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

[36] L. C. S. Motta, *Novas Abordagens para o Problema de Recobrimentos de Rotas*; Tese de Mestrado, Departamento de Ciência da Comptação, Universidade Federal Fluminense, Rio de Janeiro, Brasil, 2001.

[37] L. C. S. Motta, L. S. Ochi and C. A. J. Martinhon, *Reduction rules for the Covering Tour Problems*; In Electronic Notes in Discrete Mathematics , pp: 168-171, vol. 7, ELSEVIER, ISNN: 0166-218X, Guest Editors: J. L. Swarcfiter and S. Song; 2001.

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

[39] J. R. Oppong and M. J. Hodgson, *Spatial acessibility to health care facilities in suhum district, ghana*, Professional Geographer 46 (1994), 199-209.

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

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

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