ALGORITMOS HEURÍSTICOS E META-HEURÍSTICAS HÍBRIDAS APLICADAS AO PLANEJAMENTO DE UMA REDE DE TELECOMUNICAÇÕES COM TOPOLOGIA ANEL-ESTRELA

Thayse Christine Souza Dias

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

Aprovada por:

Prof. Márcia Helena Costa Fampa, D.Sc.

Prof. Nelson Maculan Filho, D.Sc.

Prof. Sulamita Klein, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARÇO DE 2006

DIAS, THAYSE CHRISTINE SOUZA

Algoritmos heurísticos e metaheurísticas híbridas aplicadas ao planejamento de uma rede de telecomunicações com topologia anel-estrela [Rio de Janeiro] 2006

XI, 95 p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2006)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Problema do ciclo mediano sem restrições de capacidade
2. Metaheurísticas
3. Otimização combinatória

I. COPPE/UFRJ II. Título (série)

Ao meu filho Fernando Neto e aos meus pais.

Agradecimentos

Ao meu Deus, que por sua vontade, obra e graça me permitiu cumprir mais essa etapa da minha vida.

Ao meu filho Fernando Neto, que apesar de tão pequeno soube entender a ausência da mãe e me dar forças pra continuar.

Aos meus pais por terem sempre acreditado que eu seria capaz, pelo carinho e pelo apoio que nunca me faltaram, além da disponibilidade em cuidar de Fernando Neto durante um ano. Amo muito vocês.

Aos meus irmãos por todo apoio, em especial a Thatyana pela companhia durante os dois anos que juntas enfrentamos o mestrado, o Rio de Janeiro e a ausência da família.

A cada membro da minha família por todo apoio e carinho. Em especial ao meu avô Assis (in memorian) que nos ensinou que com trabalho e dedicação se chega ao longe.

À professora Márcia Fampa, minha orientadora, por ter me aceito como sua orientanda, além da paciência e compreensão dos momentos difíceis. Muito obrigado pela confiança em minha pessoa.

Ao professor Elder Magalhães Macambira, pela sugestão do tema, confiança, paciência e orientação criteriosa ao longo de todo este trabalho. Muito obrigada por sua valiosa contribuição, incentivo e apoio na revisão deste texto. Você é um exemplo de força de vontade e dedicação.

Ao professor Lucídio dos Anjos Cabral que me orientou extra-oficialmente, muito obrigada por todas as sugestões e pela paciência nos momentos difíceis.

Aos professores Nelson Maculan Filho, Sulamita Klein e Satoru Ochi por aceitarem participar desta banca, agregando valor inestimável a este trabalho.

Ao professor Mauro Rincon, por ter me apresentado ao mundo da otimização.

Aos professores e funcionários do PESC, em especial aos professores Adilson e Gerson Zaverucha, e a Solange, por ter me ajudado à distância a resolver os muitos problemas burocráticos enfrentados.

A todos os professores da graduação em especial Carlos Alberto Braga, José Valdek, e ao professor João Marcos do Ó, por ter me apresentado ao mundo da pesquisa.

Aos meus amigos do NCE, Vinícius, Patrícia, Denise, Flávio, Michaele, Cristiane, Paula Patrícia, Roberta, Giselle e ao professor Ubiratan, por toda a amizade, companheirismo e conhecimentos adquiridos na convivência com vocês. Em especial à Juliana Pontes que sempre se mostrou uma grande amiga.

Aos meus amigos da COPPE, Alex Marin, Natanael, Karla, Fabiana, Luidi, Elivelton, Alexandre, entre tantos outros que me deram o prazer da convivência nesses quase três anos juntos.

A Gilberto pela ajuda imprescindível na implementação, devo a ele todo o êxito.

Aos meus amigos de longas datas por todo o apoio durante o tempo que estive distante, e aos amigos que conquistei na estada na cidade maravilhosa, em especial a Riane, Alexandre Bittencourt, Márcio e toda a família Calux, Eduardo, Gabi e Isabel Chabo, e tantos outros, que direta ou indiretamente, me ajudaram e me apoiaram a trilhar este caminho.

Ao meu amigo Fágner por ter me acolhido no Rio de Janeiro e por sempre ter me incentivado a estudar e a buscar cada vez mais conhecimento. A nossa amizade e convivência serão inesquecíveis.

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

ALGORITMOS HEURÍSTICOS E METAHEURÍSTICAS HÍBRIDAS APLICADAS AO PLANEJAMENTO DE UMA REDE DE TELECOMUNICAÇÕES COM TOPOLOGIA ANEL-ESTRELA

Thayse Christine Souza Dias

Março / 2006

Orientadores: Márcia Helena Costa Fampa
Elder Magalhães Macambira

Programa: Engenharia de Sistemas e Computação

O problema apresentado surge no planejamento de uma rede de telecomunicações, sendo conhecido como Problema do Ciclo Mediano sem Restrições de Capacidade. No PCMRC a meta é localizar um anel através de um subconjunto de nós com objetivo de minimizar a soma dos custos de roteamento, ou seja, o custo de ligação dos nós do anel, e de atribuição dos nós fora do anel aos nós mais próximos no anel. Foram propostas metaheurísticas híbridas Simulated Annealing + Busca Tabu (BT), GRASP + BT e GRASP + GVNS, onde experimentos computacionais compravaram a eficiência da metaheurística GRASP associada a outra metaheurística que expanda a vizinhança, assim dentre as metaheurísticas testadas a GGVNS destacou-se quanto à qualidade das soluções obtidas quando comparada às outras metaheurísticas propostas e também à técnica VNTS proposta na literatura.

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

HEURISTICS ALGORITHMS AND HYBRID METAHEURISTICS APPLIED TO DESIGN OF A TELECOMMUNICATIONS NETWORK WITH A RING-STAR TOPOLOGY

Thayse Christine Souza Dias

March / 2006

Advisors: Márcia Helena Costa Fampa
Elder Magalhães Macambira

Department: Computer and Systems Engineering

The problem presented in this work appears in the context of design of a telecommunication network and it is known as The Ring Star Problem (RSP). In the RSP the aim is to locate a simple cycle through a subset of nodes with the objective of minimizing the sum of the routing cost, or either, the cost of linking of nodes of the ring, and of assignment costs of the nodes not in the ring to their closest node on the ring. It was proposed hybrid metaheuristic as Simulated Annealing + Tabu Search (TS), GRASP + TS and GRASP + GVNS where the computational experiments demonstrated the efficiency of GRASP combined with other metaheuristics that expand the neighbourhood thus among the evaluated metaheuristics the GRASP + GVNS overcame, in terms of quality solution, other metaheuristics proposed and the VNTS approach.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O mundo das telecomunicações está em franca expansão e associada a essa expansão surge a necessidade de um planejamento mais adequado das redes de telecomunicações. Durante este planejamento podem aparecer diversos problemas de otimização combinatória. Alguns destes problemas se mostram bastante desafiadores, pois envolvem questões como roteamento, dimensionamento, síntese e topologia. A literatura mais recente mostra alguns bons exemplos ver [18, 38]

A definição da topologia dessas redes é um dos grandes problemas enfrentados pelas empresas, algumas fazem uso de uma topologia anel, que nas redes de comunicação de fibra ótica, possibilita a prevenção da perda de conexão durante a transmissão dos dados, garantindo assim um serviço de comunicação contínua entre os clientes. Outras empresas trabalham com uma topologia multi-camadas. Entretanto, este último tipo de topologia representa apenas uma parcela de toda a rede global.

Uma rede de telecomunicações é constituída basicamente de dois níveis: a rede de acesso local e a rede principal (ou *backbone*). A rede de acesso local conecta os terminais ou (nós usuários) a falsos concentradores (*switches* ou

multiplexadores) enquanto que uma rede principal tem como função interconectar esses concentradores ou os conectar a uma unidade central (nó raiz ou depósito).

Uma das soluções adotadas para a definição topologia de algumas redes de telecomunicações é conectar os terminais aos concentradores através de conexões ponto-a-ponto, o que resulta em uma topologia estrela, e interconectar os concentradores entre si através de uma estrutura que faz uso de anéis, o que resulta em uma topologia anel. Outros exemplos de modelos adotados na definição da topologia de uma rede de telecomunicações podem ser encontrados em [18, 26].

Também podemos encontrar esse modelo de rede em anel no contexto de problemas de roteamento de veículos, tais como linhas de metrô circular e rodovias, onde temos o conjunto de paradas potenciais e todas as distâncias entre os pares de paradas. Existe uma parada específica e fixa que é o depósito de um veículo, e cada parada potencial tem seu valor específio que representa a população que tal parada serviria. O objetivo é selecionar um conjunto de paradas incluindo o depósito e determinar uma rota circular através dele, o que nos daria a estrutura anel, assim como também minimizar o custo do percurso limitando a média do custo de acessibilidade para a população servida não diretamente. Outras aplicações de modelos de rede em anel podem ser vistas com mais detalhes em [18, 40, 30].

É neste contexto de roteamento que surge o Problema do Ciclo Mediano (maiores detalhes [27]). Em [34], o Problema do Ciclo Mediano (PCM) foi estudado no contexto de localização e distribuição no planejamento de sistemas de trânsito rápido. Neste trabalho, os autores resolveram o PCM de forma heurística. Em [28] foi desenvolvido um modelo de programação linear inteira para o caso do PCM no projeto de redes de telecomunicações. Neste

trabalho, os autores propuseram a resolução do Problema de Ciclo Mediano através de um algoritmo branch-and-cut. O problema que estudaremos nesta dissertação é um caso especial do Problema do Ciclo Mediano. Em particular, o problema estudado não apresenta restrições de capacidade no ciclo mediano da rede de telecomunicações. Este problema é conhecido na literatura como Problema do Ciclo Mediano sem Restrição de Capacidade (*Ring Star Problem*).

O Problema do Ciclo Mediano sem Restrição de Capacidade (PCMRC) surge no contexto de redes de telecomunicações onde os nós usuários, também denominados de terminais, devem ser conectados a falsos concentradores numa rede principal, e estes falsos concentradores devem estar ligados a uma unidade central (nó raiz) através de um ciclo. Em outras palavras, o PCMRC consiste em selecionar, dentre um conjunto de nós usuários, um subconjunto onde serão instalados os concentradores, e interconectá-los em uma rede principal por meio de um ciclo, o que resultará numa topologia anel. Em seguida deve-se atribuir os nós usuários restantes aos concentradores através de conexões ponto-a-ponto, o que resultará numa topologia estrela para a rede de acesso local, de modo que encontremos uma solução viável que minimize o custo de todas as ligações no anel (rede principal), que chamaremos de custo de roteamento, e todas as atribuições dos nós usuários aos concentradores (rede de acesso local), que chamaremos de custo de atribuição. Ou seja, nosso objetivo é minimizar o custo do planejamento de uma rede de telecomunicações seguindo uma topologia anel-estrela.

1.1 Motivação e objetivos do trabalho

A motivação inicial para o estudo do Problema do Ciclo Mediano sem Restrições de Capacidade deve-se ao avanço crescente na área de telecomunicações e da possibilidade do emprego desse problema na redução dos custos do planejamento de uma rede de telecomunicações.

Outra motivação é devido à complexidade inerente a alguns problemas combinatórios, classificados como, problemas NP-difíceis, como é o caso do PCMRC. Nestes casos, muitos deles ainda não foram resolvidos de forma satisfatória.

Baseado nesses dois aspectos, um dos principais objetivos deste trabalho é resolver de forma heurística o Problema do Ciclo Mediano sem Restrições de Capacidade procurando manter um compromisso entre a qualidade da solução e o tempo computacional, mesmo para instâncias de tamanho moderado.

Para obter este compromisso propomos e desenvolvemos algumas metaheurísticas e estratégias de diversificação com o intuito de encontrarmos resultados melhores ou tão bons quanto os já existentes na literatura, verificação essa que será feita ao fim desse trabalho numa análise comparativa entre os métodos abordados.

1.2 Organização da dissertação

Esta dissertação está organizada em sete capítulos. Os tópicos a serem cobertos em cada um deles são descritos em seguida. Neste primeiro capítulo, será feita uma breve descrição do problema que trataremos, os principais objetivos deste trabalho, o que nos motivou para o seu desenvolvimento e a forma de abordagem empregada na resolução do problema. No segundo

capítulo apresentaremos a definição formal, a complexidade e um modelo matemático para o PCMRC bem como alguns trabalhos relacionados, como por exemplo algoritmos heurísticos e exatos empregados na resolução do PCMRC.

No terceiro capítulo, apresentaremos algumas técnicas heurísticas utilizadas para a resolução de problemas de otimização combinatória procurando ressaltar suas principais características. Em particular, nos deteremos apenas àquelas que estão relacionadas com o nosso trabalho, como por exemplo: Simulated Annealing, Busca Tabu, GRASP e VNS.

No quarto capítulo, apresentaremos uma descrição das estruturas de vizinhança e das metaheurísticas básicas avaliadas para a resolução do PCMRC.

Não satisfeitos com o uso das metaheurísticas básicas decidimos trabalhar com técnicas híbridas para a resolução do PCMRC. Assim, no quinto e sexto capítulos descreveremos algumas dessas técnicas híbridas propostas para o PCMRC. Em cada um destes capítulos, descreveremos os resultados computacionais obtidos com o emprego destas técnicas bem como uma análise comparativa com outros algoritmos existentes na literatura para o PCMRC.

Aplicadas todas essas técnicas acima, no Capítulo 6, apresentaremos uma comparação entre os resultados computacionais obtidos com a melhor destas técnicas exploradas e uma metaheurística existente na literatura para o PCMRC.

Por último, no capítulo 7 mostraremos as conclusões envolvendo as técnicas estudadas, os algoritmos apresentados, resultados obtidos e propostas para trabalhos futuros.

1.3 Conceitos preliminares

Nessa seção procuraremos deixar claro alguns conceitos e notações que empregaremos comumente no decorrer desta dissertação. Os primeiros conceitos a serem apresentados são referentes à Teoria dos grafos.

1.3.1 Definições em Teoria dos grafos

As definições apresentadas aqui foram compiladas de [6, 5, 4]

Definição 1.1. *Um grafo simples $G = (V, E)$ consiste de um conjunto finito não vazio V de elementos chamados vértices e um conjunto de pares não-ordenados de elementos distintos de V, chamados arestas, isto é, $E \subseteq \{(u, v) : u, v \in V, \text{ com } u \neq v\}$.*

Em geral convenciona-se que $|V| = n$ e $|E| = m$, ou seja, o número de vértices é igual a n e de arestas igual a m, respectivamente.

Definição 1.2. *Um grafo direcionado (digrafo) é um par ordenado $D = (V, A)$, onde V é um conjunto finito, não vazio de elementos denominados vértices e A é um conjunto de pares ordenados de vértices distintos de V, denominado de arestas direcionadas ou arcos: $[v, w] \neq [w, v]$.*

Os conceitos apresentados logo abaixo são referentes a grafos simples $G = (V, E)$.

Definição 1.3. *Se $e = (u, v) \in E$ é uma aresta dizemos que e incide em u e v, que u e v são seus extremos e que u e v são adjacentes.*

Definição 1.4. *Seja $v \in V$ um vértice qualquer em G. O grau do vértice v corresponde ao número de arestas incidentes a v.*

Denotaremos $\delta(v)$ o conjunto de todas as arestas incidentes ao vértice v. Assim, o menor grau de um vértice v pode ser denotado por $|\delta(v)|$.

Caso tenhamos mais de um grafo, por exemplo, G, H e W, denotaremos seus conjuntos de vértices por $V(G)$, $V(H)$ e $V(W)$, respectivamente; e o conjunto de arestas por $E(G)$, $E(H)$ e $E(W)$, respectivamente.

Definição 1.5. *Um grafo H é dito ser um subgrafo de um grafo G se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$.*

Definição 1.6. *Se $V(H) = V(G)$ então H é um subgrafo gerado ou espalhamento de G.*

Definição 1.7. *Um subgrafo H de G é dito ser um subgrafo induzido de G quando $\forall v, w \in V(H)$ se $(v, w) \in E(G)$ então $(v, w) \in E(H)$.*

Dado um conjunto $V' \subseteq V$, denotamos por $E(V')$ o conjunto de todas as arestas com ambas as extremidades dos vértices em V', isto é, $E(V') = \{(u, v) \in E : u, v \in V'\}$. O corte $\delta(V')$ corresponde ao conjunto de todas as arestas que são incidentes a exatamente um vértice em V', ou seja, $\delta(V') = \{(u, v) \in : u \in V' \text{ e } v \notin V'\}$

Definição 1.8. *Seja $G = (V, E)$ um grafo. Se para quaisquer dois vértices $u, v \in V$, com $u \neq v$, temos u e v adjacentes então G é completo. O grafo completo com n vértices é denotado por $K_n = (V_n, E_n)$.*

Definição 1.9. *Um caminho em um grafo $G = (V, E)$ é uma sequência não-vazia $P = \{e_1, e_2, \ldots, e_k\}$ de arestas, onde $e_i = (v_i, v_{i+1}) \in E$ e $v_i \neq v_j$ para $i \neq j$.*

Definição 1.10. *Um ciclo é um caminho $v_1, \ldots, v_k, v_{k+1}$ onde $v_k = v_{k+1}$ e $k \geq 3$.*

Definição 1.11. *Um ciclo hamiltoniano em um grafo $G = (V, E)$ é um ciclo que contém todos os vértices de G.*

Definição 1.12. *Um grafo é conexo se possui um caminho entre dois quaisquer de seus vértices, caso contrário, é dito ser desconexo.*

Definição 1.13. *Seja $G = (V, E)$ um grafo. Dizemos que H é um componente de G se H é um subgrafo conexo maximal em G.*

Definição 1.14. *Uma árvore é um grafo conexo, sem ciclos.*

Definição 1.15. *Uma floresta é um grafo onde cada componente é uma árvore.*

Capítulo 2

Problemas de localização de ciclos medianos

2.1 Introdução

Neste capítulo apresentaremos alguns problemas de otimização combinatória que podem ser caracterizados como problemas de localização de ciclos medianos, dentre eles o Problema do Ciclo Mediano sem Restrições de Capacidade (PCMRC). Além disso, apresentaremos duas formas de abordagens heurísticas e exata, que podem ser empregadas na resolução do PCMRC.

Podemos encontrar aplicações desses problemas em diversas áreas, tais como em telecomunicações, roteamento de veículos e localização, nesse capítulo veremos algumas dessas aplicações.

O problema descrito em [27] consiste em determinar um ciclo único através de um subconjunto de vértices de um grafo envolvendo dois tipos de custo: um de roteamento e outro de atribuição. Esse problema é denominado Problema do Ciclo Mediano e pode ser descrito como segue.

Consideremos um grafo completo misto (formado por arcos e arestas) $G = (V, E \cup A)$, onde $V = \{v_1, v_2, \ldots, v_n\}$ é um conjunto de vértices, que inicialmente chamaremos de nós usuários, $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ representa o conjunto das arestas, $A = \{[v_i, v_j] : v_i, v_j \in V\}$ é o conjunto dos arcos, onde também estão incluídos os laços $[v_i, v_i]$, e o vértice v_1 é dito ser o nó raiz.

A cada aresta (v_i, v_j), temos um custo de roteamento $c_{v_i,v_j} > 0$ a ela associado, como também a cada arco $[v_i, v_j]$, temos um custo de atribuição $d_{v_i,v_j} > 0$ associado. Assumimos $n \geq 6$ a fim de eliminarmos instâncias triviais e degeneradas.

Uma solução viável para o Problema de Localização do Ciclo Mediano consiste em determinar um subconjunto $V' \subset V$, incluindo o depósito v_1 e pelo menos outros dois vértices. Este subconjunto V' dará origem a um ciclo enquanto que os demais vértices darão origem a uma estrutura em estrela. Escolheremos os vértices pertencentes a V' que interconectados através das arestas sujeitos a um custo de roteamento c_{v_i,v_j}, constituirão o nosso ciclo único, e os vértices restantes serão conectados aos vértices do ciclo, sujeitos a um custo de atribuição d_{v_i,v_j}. O custo de roteamento de uma solução é dado pela soma de todos os custos de roteamento das arestas no ciclo, e o custo de atribuição de uma solução visitando V' é definida como a soma dos mínimos custos de atribuição dos vértices não-visitados aos vértices visitados, ou seja,

$$\sum_{v_i \in V \setminus V'} \min_{v_j \in V'} d_{v_i,v_j}.$$

Uma das versões do PCM consiste em encontrar um custo de roteamento mínimo, tal que, o custo de atribuição seja no máximo igual a d_0. Esta versão é denominada Problema do Ciclo Mediano com Restrições de Capacidade. Uma outra versão consiste em determinar um ciclo, tal que, a soma dos custos

de roteamento e de atribuição seja mínimo. Esse problema é denominado de Problema do Ciclo Mediano sem Restrições de Capacidade e será o que abordaremos nessa dissertação.

Em seguida, aparesentaremos algumas aplicações e problemas relacionados com o Problema do Ciclo Mediano.

2.2 Aplicações e problemas relacionados

Uma aplicação dos problemas de localização de ciclos medianos é a localização de uma infra-estrutura de transporte em forma circular, como as linhas de metrô. Duas outras aplicações podem ser vistas na modelagem de rotas de entrega ou coleta através de um conjunto de facilidades, e na localização de caixas de correios.

No contexto de telecomunicações, podemos citar a modelagem de uma rede do tipo descrita por Xu e outros [40], onde um conjunto de nós usuários (ou *hubs*) devem ser interligados por um anel, e os demais nós devem ser atribuídos aos *hubs*, procurando minimizar o custo total e garantir uma dada confiabilidade.

Como visto na seção anterior, na função objetivo do Problema do Ciclo Mediano com Restrições de Capacidade, apenas os custos de roteamento são considerados mas, um limite é imposto no custo de atribuição. Já o Problema do Ciclo Mediano sem Restrições de Capacidade consiste da minimização da soma dos custos de roteamento e atribuição. Problemas similares surgem na modelagem de rotas de coleta e entrega através de um conjunto de locações. Labbé e Laporte [25] consideraram o caso de localizar convenientemente caixas de correios levando em conta custo de coleta e coveniência

do usuário. Uma outra aplicação similar é na localização de coletores de resíduos recicláveis numa cidade.

Um caso importante do Problema do Ciclo Mediano sem Restrições de Capacidade é o Problema do Caixeiro Viajante obtido quando $d_0 = 0$. Muitos autores têm estudado a localização de um ciclo simples num grafo não-direcionado sobre diferentes objetivos e restrições que aquelas do Problema do Ciclo Mediano sem Restrições de Capacidade. Abaixo descreveremos alguns destes problemas:

Problema do Caixeiro Viajante Generalizado: (Fischetti, Salazar and Toth [10, 11]) : considere um grafo não-direcionado com o conjunto de vértices dividido em agrupamentos (ou *clusters*). Cada aresta tem um custo associado. O Problema do Caixeiro Viajante Generalizado deseja determinar um ciclo único de custo mínimo que visite cada agrupamento pelo menos uma vez. Uma outra versão do Problema do Caixeiro Viajante Generalizado procura encontrar um ciclo único, de custo mínimo, que visite cada agrupamento exatamente uma vez.

Problema de Orientação (Fischetti, Salazar and Toth [12]; Gendreau, Laporte, Semet [15]) : considere um grafo não-direcionado onde cada aresta tem um custo e cada vértice tem um prêmio associado. O Problema de Orientação consiste em encontrar um ciclo único de um subconjunto do conjunto de vértices, com limites nos custos, que maximize o prêmio total coletado nos vértices visitados.

Problema do Caixeiro Viajante com Coleta de Prêmios (Balas [1]) : seja um grafo não-direcionado onde cada aresta tem um custo e cada vértice tem um prêmio associado. O Problema do Caixeiro Viajante com Coleta de Prêmios deseja determinar um ciclo de custo mínimo de um subconjunto do

conjunto de vértices que colete pelo menos uma dada quantia de prêmios nos vértices visitados.

Problema de Ciclo (Bauer [3]) : considere um grafo não-direcionado onde cada aresta tem um custo associado. O problema de ciclo exige encontrar um ciclo único de custo mínimo, não necessariamente hamiltoniano, no grafo.

Problema de Recobrimento de Rotas (The Covering Tour Problem) (Gendreau, Laporte, Semet [15]) : considere um grafo não-direcionado com um conjunto de vértices $V = U \cup W$. Cada aresta tem um custo associado a ela. O Problema de Recobrimento de Rotas consiste em determinar um ciclo de custo mínimo num subconjunto de U, tal que, todo vértice de W está coberto pelo ciclo.

2.3 Problema do ciclo mediano sem restrições de capacidade

O problema de otimização combinatória conhecido Problema do Ciclo Mediano sem Restrições de Capacidade pode ser formalmente definido como segue. Consideremos uma rede de telecomunicações genérica, que representaremos por um grafo completo misto $G = (V, E \cup A)$, onde $V = \{v_1, v_2, \ldots, v_n\}$ é um conjunto de vértices, que inicialmente chamaremos de nós usuários, $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ representa um conjunto de arestas e $A = \{[v_i, v_j] : v_i, v_j \in V\}$ é o conjunto dos arcos, onde também estão incluídos os laços $[v_i, v_i]$. O vértice v_1 é dito raiz. Cada aresta (v_i, v_j), possui um custo de roteamento $c_{v_i,v_j} > 0$ associado a ela, como também cada arco $[v_i, v_j]$, possui um custo de atribuição $d_{v_i,v_j} > 0$ associado a ele. O Problema do Ciclo Mediano sem Restrições de Capacidade consiste em

determinar um ciclo hamiltoniano através de um subconjunto V' de V, incluindo v_1 e pelo menos dois outros vértices, que minimize a soma do custo de roteamento do anel e o custo de atribuição dos vértices fora do anel aos vértices mais próximos no anel. O custo de roteamento da solução é a soma de todos os custos das arestas no anel. O custo de atribuição é calculado como a soma dos mínimos custos de atribuição dos vértices não visitados aos vértices visitados, ou seja,

$$\sum_{v_i \in V \setminus V'} \min_{v_j \in V'} d_{v_i,v_j}. \tag{2.1}$$

Dois problemas que surgem na área de telecomunicações, cujo cenário são estruturas em ciclo, e que possuem uma relação estreita com o PCMRC são:

- Problema do Ciclo Steiner Mediano sem Restrição de Capacidade, e
- Problema dos m-ciclos medianos capacitados

O Problema do Ciclo Steiner Mediano sem Restrição de Capacidade (ver [30]) difere do PCMRC pelo fato de apresentar um conjunto adicional de vértices W, nós de Steiner, e por usar a equação (2.2) para o custo de atribuição d_{v_i,v_j} ao invés da equação (2.1):

$$\sum_{v_i \in W} \min_{v_j \in V'} d_{v_i,v_j}. \tag{2.2}$$

Já o Problema dos m-ciclos medianos capacitados (ver [2]), difere do PCMRC por apresentar mais de um anel, na topologia da rede, e por cada anel possuir um número máximo de vértices que podem ficar ligados nos anéis.

2.3.1 Problema de decisão e complexidade

Nesta seção apresentaremos um resultado sobre a complexidade do PCMRC. Em particular, é mostrado a $NP-completude$ do PCMRC.

Considere o seguinte problema de decisão abaixo:

Instância: dado um grafo completo $K_n = (V_n, E_n)$ com custo de roteamento $c_{v_i,v_j} \geq 0$ associado a cada aresta $(v_i, v_j) \in E_n$ um custo de atribuição $d_{v_i,v_j} \geq 0$ associado a cada par de vértices $v_i, v_j \in V_n$ e um inteiro B.

Questão: existe um ciclo em G, tal que, o custo de roteamento mais o custo de atribuição seja menor ou igual a B?

Proposição: [32] O problema de decisão do PCMRC é $NP-completo$.

Pode-se verificar que o PCMRC é um problema $NP-completo$, visto que o Problema do Caixeiro Viajante é um caso especial dele. De fato, se para cada par de vértices v_i e v_j o custo de atribuição é definido como

$$d_{v_i,v_j} = \begin{cases} \infty, & \text{se } v_i \neq v_j, \\ 0, & \text{caso contrário.} \end{cases}$$

então o PCMRC torna-se um PCV.

2.3.2 Formulação matemática (Modelo)

O Problema do Ciclo Mediano sem Restrições de Capacidade pode ser modelado segundo [[28]] como um problema de programação linear inteira mista $0-1$ como segue-se:

para cada aresta $[v_i, v_j] \in E$, seja x_{v_i,v_j} uma variável binária, tal que,

$$x_{v_i,v_j} = \begin{cases} 1, \text{ se, e somente se } (v_i, v_j) \text{ aparece no ciclo,} \\ 0, \text{ caso contrário.} \end{cases}$$

Para cada arco $[v_i, v_j] \in A$, seja y_{v_i,v_j} uma variável binária, tal que,

$$y_{v_i,v_j} = \begin{cases} 1, \text{se, e somente se, o vértice } v_i \text{ é atribuído ao vértice } v_j \text{ no ciclo,} \\ 0, \text{ caso contrário.} \end{cases}$$

Observe que, se um vértice v_i estiver no ciclo então ele é atribuído a ele mesmo, ou seja, $y_{ii} = 1$. Além disso, assuma que $S \subset V$ seja um conjunto de vértices como definido no capítulo 1 podemos ter:

- $E(S) = \{(v_i, v_j) \in E : v_i, v_j \in S\}$,
- $\delta(S) = \{(v_i, v_j) \in E : v_i \in S,\ v_j \notin S\}$. Se $S = \{v_i\}$, então escreveremos $\delta(i)$, ao invés de $\delta(\{v_i\})$.

Para $E' \subset E$, definimos $x(E') = \sum_{[v_i,v_j] \in E'} x_{v_i,v_j}$, que representa a soma das arestas dos conjunto E'.

Podemos então representar o modelo matemático para o PCMRC como sendo:

$$Min \sum_{(v_i,v_j) \in E} c_{v_i,v_j} x_{v_i,v_j} + \sum_{[v_i,v_j] \in A} d_{v_i,v_j} y_{v_i,v_j} \tag{2.3}$$

sujeito a:

$$x(\delta(i)) = 2y_{v_i,v_i}, \quad \forall v_i \in V, \tag{2.4}$$

$$x(\delta(S)) \geq 2 \sum_{v_j \in S} y_{v_i,v_j}, \quad \forall S \subset V : v_1 \notin S,\ v_i \in S, \tag{2.5}$$

$$\sum_{v_j \in V} y_{v_i,v_j} = 1 \quad \forall v_i \in V \setminus \{v_1\}, \tag{2.6}$$

$$y_{v_i,v_j} \leq y_{v_j,v_j}, \quad \forall [v_i, v_j] \in A,\ v_i \neq v_j, \tag{2.7}$$

$$y_{v_1,v_1} = 1, \tag{2.8}$$

$$y_{v_1,v_j} = 0, \quad \forall v_j \in V \setminus \{v_1\}, \tag{2.9}$$

$$y_{v_i,v_j} \geq 0 \quad \forall [v_i, v_j] \in A, \tag{2.10}$$

$$y_{v_j,v_j} \in \{0,1\}, \quad \forall v_j \in V \setminus \{v_1\}, \tag{2.11}$$

$$x_{v_i,v_j} \in \{0,1\}, \quad \forall (v_i,v_j) \in E. \tag{2.12}$$

Na função objetivo (2.3) temos a minimização do custo total das conexões, considerando todas as arestas e todos os arcos.

As restrições (2.4) são denominadas restrições de grau e estabelecem que o grau de v_i é igual a 2, se e somente se ele pertence ao ciclo (isto é, $y_{v_i,v_i} = 1$).

As restrições (2.5) são denominadas restrições de conectividade e elas estabelecem que $S \subseteq V \setminus \{v_1\}$ deve estar conectado ao seu complemento por pelo menos duas arestas do ciclo sempre que pelo menos um vértice $v_i \in S$ é visitado pela solução.

As restrições (2.6) estabelecem que ou o vértice v_i está no ciclo e assim $y_{v_i,v_i} = 1$, ou ele está atribuído a um vértice v_j no ciclo, assim, $y_{v_i,v_i} = 0$ e $y_{v_i,v_j} = 1$. Essas restrições são denominadas restrições de atribuição.

As restrições (2.7) estabelecem que um vértice $v_i \in V$ pode ser atribuído a outro vértice $v_j \in V$ somente se v_j está no ciclo $\left(y_{v_j,v_j} = 1\right)$. As restrições (2.8) e (2.9) impõem que o vértice v_1 está na solução. A combinação das restrições (2.4), (2.8), (2.9) e (2.12) garante que a solução percorrerá no mínimo um ciclo com o depósito v_1.

As restrições (2.5) limitam o número de ciclos a um, e combinando (2.4), (2.6), (2.7) e (2.10) garante-se que todo vértice que não pertence ao ciclo é atribuído a um vértice no ciclo. As restrições (2.11) e (2.12) correspondem as restrições de integralidade das variáveis x_{v_i,v_j} e y_{v_i,v_j}. As condições de integra-lidade nas variáveis y_{v_i,v_j}, com $v_i \neq v_j$ são desnecessárias, pois para um dado vetor inteiro x, o objetivo é minimizado quando um vértice fora do ciclo é atribuído ao vértice mais próximo no ciclo $\left(y_{v_i,v_j} = 1\right)$.

Implicitamente, o modelo matemático impõe que o ciclo visite pelo menos três vértices, incluindo o depósito. Isto garante que a comunicação entre os vértices não é interrompida mesmo se um vértice da rede principal falhar.

2.3.3 Trabalhos relacionados

Nessa seção apresentaremos alguns trabalhos relacionados ao Problema do Ciclo Mediano sem Restrições de Capacidade. Dividiremos esse estudo seguindo duas formas de abordagem que podem ser empregadas na solução de problemas de otimização combinatória: heurística e exata.

Abordagem heurística

O Problema do Ciclo Mediano sem Restrições de Capacidade foi estudado inicialmente por Labbé e outros em [27]. Após esse primeiro estudo, alguns trabalhos foram publicados na literatura envolvendo a resolução do PCMRC de forma heurística. A seguir, apresentaremos uma breve descrição destes trabalhos.

Variable Neighborhood Tabu Search Perez e outros [34] projetaram uma heurística híbrida denominada *Variable Neighborhood Tabu Search* (VNTS), para a resolução do PCMRC. Esta heurística consiste na combinação das metaheurísticas *Variable Neighbourhood Search* e *Tabu Search*. Assim, através desta combinação, os autores conseguiram uma heurística composta de duas fases: uma construtiva e outra que engloba uma série de buscas locais que fazem uso de técnicas tabu. Quando a busca local pára num mínimo local não-tabu, um procedimento de perturbação inicia uma nova busca local. Tanto na busca local, onde o melhor movimento possível é aplicado até que nenhum movimento melhor exista, quanto no procedimento de perturbação,

onde aplica-se um número de movimentos aleatórios, os movimentos padrões usados para gerar soluções vizinhas foram adição *(add)*, exclusão *(drop)* e troca de vértices *(swap)*. Como o PCV é um caso especial do PCMRC, os autores também fizeram uso de ferramentas tradicionais do PCV, como por exemplo, 2-opt, 3-opt e o procedimento *Lin-Kernighan* [23, 31]. Os autores testaram a metaheurística proposta para um conjunto de instâncias do Caixeiro Viajante envolvendo entre 50 e 200 vértices e tendo formato EUC2D (distâncias euclidianas).

MGA e RKEA Em [35] os autores propuseram duas heurísticas para o Problema do Ciclo Mediano sem Restrições de Capacidade. A primeira heurística, denominada de *Multistart Greedy Add* (MGA), é constituída de duas fases: uma fase de construção *multistart* gulosa e uma fase de busca local. Na primeira fase constroí-se um ciclo composto de um número limitado de vértices escolhidos aleatoriamente e aumenta-se o ciclo iterativamente adicionando-se o vértice que produzir uma maior redução no custo. Após a construção da solução inicial, a segunda fase da heurística é chamada. Esta fase consiste em aplicar repetidamente um número de rotinas de melhoria até que a solução atual não possa ser melhorada. A segunda heurística, denominada *Random Keys Evolutionary Algorithm* (*RKEA*) é um algoritmo evolucionário que combina as soluções obtidas na heurística MGA com o intuito de obter melhores soluções para o PCMRC. Esta metaheurística faz uso do mecanismo de chaves aleatórias e de procedimentos básicos dos algoritmos evolucionários, tais como crossover e geração de populações. Além destes procedimentos, uma fase de intensificação é empregada na melhor solução obtida através da heurística Lin-Kernighan [23, 31]. As instâncias utilizadas foram

as do Caixeiro Viajante variando entre 51 e 200 vértices como descritas em [[27],[28]].

Abordagem Exata

Problemas de ciclos medianos Como já foi mencionado anteriormente o Problema do Ciclo Mediano sem Restrições de Capacidade foi formulado por Labbé e outros [27] como sendo um problema de programação linear inteira mista $0-1$. Neste trabalho, os autores realizaram um estudo poliédrico do PCMRC e propuseram novas desigualdades válidas para o politopo. Através destas desigualdades, foi desenvolvido um algoritmo *branch-and-cut* para a resolução exata do PCMRC. Para maiores detalhes sobre o algoritmo e as desigualdades propostas para o PCMRC, sugerimos uma leitura no artigo [27]. Os autores testaram o algoritmo usando três classes de instâncias, na primeira classe foram testadas instâncias do Caixeiro Viajante envolvendo 50 até 200 vértices, e usando a distância Euclidiana no cálculo das distâncias. Na segunda classe as instâncias foram geradas segundo $c_{v_i,v_j} = \alpha \times l_{v_i,v_j}$ e $d_{v_i,v_j} = (10-\alpha) \times l_{v_i,v_j}$, onde $\alpha \in 3,5,7,9$ e l_{v_i,v_j} correspondendo a distância Euclidiana entre os vértices v_i e v_j. Na terceira classe de instâncias usou-se $l_{v_i,v_j} := c_{v_i,v_j} := d_{v_i,v_j}$, e os custos d_{v_i,v_j} foram gerados aleatoriamente no intervalo $[0,1000]$ para todo $v_i \in V$.

Após este algoritmo *branch-and-cut*, alguns algoritmos deste gênero foram propostos para o PCMRC. A diferença entre eles consistia no uso do modelo matemático e em quais desigualdades válidas empregar no algoritmo *branch-and-cut*. Para maiores detalhes sobre esses algoritmos sugerimos [13, 29].

Capítulo 3

Técnicas heurísticas de otimização

Na computação, pesquisa operacional, engenharias, matemática, entre outras áreas, encontramos problemas considerados intratáveis, que dizemos pertencerem as classes NP-completo ou NP-difícil, uma vez que não existem algoritmos conhecidos com complexidade polinomial para resolvê-los.

Resolver esses problemas considerados intratáveis de maneira exata nem sempre é possível devido aos requisitos de tempo, decorrentes de um aumento exponencial no número de operações necessárias para resolvê-los em relação ao tamanho da entrada, mesmo com os avanços tecnológicos em termos de capacidade e velocidade de processamento. Para instâncias pequenas dos problemas intratáveis, existem algoritmos capazes de encontrar uma solução ótima, mas à medida que o tamanho das instâncias aumentam os algoritmos já não conseguem encontrar uma solução ótima num tempo computacional razoável.

Devido às dificuldades mencionadas acima, procura-se desenvolver algoritmos que encontrem soluções aproximadas (próximas ao ótimo), mesmo sem

garantir a otimalidade, nem quão próximo a solução obtida está do ótimo, a um custo computacional razoável. Tais algoritmos são denominados de heurísticas.

O desafio dos pesquisadores têm sido produzir, em tempo mínimo, soluções cujos valores sejam tão próximas quanto possível do valor ótimo, dessa forma, os esforços nesse sentido têm sido grandes, e heurísticas muito eficientes foram desenvolvidas para diversos problemas. Porém, a maioria dessas heurísticas desenvolvidas é muito específica para um problema particular, não garantindo a eficiência ou até mesmo a aplicabilidade para uma classe mais ampla de problemas.

Objetivando desenvolver heurísticas de caráter mais geral que sejam eficientes na solução de uma classe mais ampla de problemas, foram propostas uma série de heurísticas genéricas, conhecidas como metaheurísticas, que reúnem conceitos das próprias heurísticas clássicas, da evolução biológica, da inteligência artificial, da física, de sistemas neurais entre outros.

Uma metaheurística é composta por dois tipos de parâmetros: parâmetros gerais que estão presentes em qualquer algoritmo do problema considerado e parâmetros de calibragem que devem ser configurados para cada tipo de pro-blema. Nas próximas seções trataremos as metaheurísticas que utilizaremos nessa dissertação, conhecidas como, *simulated annealing*, que baseia-se numa analogia com processos termodinâmicos, busca tabu, que faz uso de uma memória flexível para tornar o processo de busca mais eficaz, *Greedy Randomized Adaptive Search Procedure*, que é composto por duas heurísticas, uma heurística construtiva (método guloso) e uma heurística de refinamento (procedimento de busca local) e por último, *Variable Neighborhood Search*, que consiste em usar várias estruturas de vizinhança durante a busca local.

3.1 Métodos construtivos

Os métodos construtivos, ou heurísticas construtivas, selecionam seqüencialmente elementos de um conjunto finito, eventualmente descartando alguns já selecionados, de tal forma que ao final obtenha-se uma solução viável.

No caso de termos um método de inserção, cada elemento a ser inserido, varia de acordo com a função de avaliação (ou objetivo) adotada, esta por sua vez depende do problema abordado. Geralmente, nas heurísticas clássicas, adota-se uma função gulosa para ordenação dos elementos, e essa função estima o benefício de inserção de cada elemento, e somente o "melhor" elemento é inserido a cada passo. Temos que diferentes funções de avaliação conduzem a diferentes soluções finais.

Uma outra maneira muito comum de se gerar a solução inicial é construí-la de maneira aleatória, ou seja, a cada passo um elemento a ser inserido na solução é selecionado aleatoriamente dentre um conjunto de elementos ainda não-selecionados. Essa escolha aleatória possibilita uma simplicidade maior na implementação, porém poderá produzir soluções finais de baixa qualidade, o que poderá exigir um maior esforço na fase de busca local.

3.2 Métodos de busca local

Os métodos de busca local, ou heurísticas de refinamento, constituem uma família de técnicas baseadas na noção de vizinhança, isto é, seja S um espaço de solução de um problema de otimização combinatória; f a função objetivo a minimizar (ou maximizar). Considere uma função N, que depende da estrutura do problema tratado, e associa a cada solução $s \in S$, sua vizinhança $N(s) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de s. A modificação

m que transforma uma solução s em outra s', que esteja em sua vizinhança, é chamada movimento, e a operação é representada por $s' \leftarrow s \oplus m$.

Em outras palavras, as heurísticas de refinamento partem de uma solução inicial viável, que pode ser obtida por uma heurística construtiva ou gerada aleatoriamente, e a cada iteração vai percorrendo a vizinhança adotada. Um problema crítico desse método é a escolha da estrutura de vizinhança ou a forma na qual ela será definida.

Dentre os métodos de busca local conhecidos podemos citar: os métodos de descida/subida, método primeiro de melhora e os métodos randômicos de descida/subida. Para maiores detalhes sobre esses métodos ver [39].

3.3 Metaheurísticas

As metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente uma solução ótima. Consistindo na aplicação, em cada passo de uma heurística subordinada, a qual tem que ser modelada para cada problema específico.

Como dito anteriormente, ao contrário das heurísticas convencionais, as metaheurísticas são de caráter geral e têm condições de escapar de ótimos locais ainda distantes de um ótimo global. As metaheurísticas, assim como os métodos de busca locais tradicionais, caracterizam-se entre si basicamente pelas seguintes aspectos:

a) critério de escolha de uma solução inicial;

b) definição de uma vizinhança $N(s)$ de uma solução s;

c) critério de seleção de uma solução vizinha dentro de $N(s)$;

d) critério de término (ou parada).

3.3.1 *Simulated annealing* (SA)

A metaheurística *simulated annealing* (SA) é derivada do processo de temperagem física de certos materiais, que consiste em submetê-los inicialmente a altas temperaturas e reduzí-las gradualmente até atingirem, com aumentos e reduções do estado de energia, o equilíbrio térmico; tornando-os assim, consistentes e rígidos. Esse estado do material refere-se à configuração mínima de seus átomos, e tal configuração corresponde ao valor mínimo da função objetivo de um problema de otimização. Essa técnica foi proposta por Kirkpatrick e outros em [24] numa analogia aos processos físicos de resfriamento na área de Mecânica Estatística dos materiais sólidos.

Essa metaheurística gera sequências de estados do material caracterizados pelas posições de suas partículas; de forma que, sob altas temperaturas estão totalmente "desorganizadas" correspondendo a uma solução aleatória s' vizinha da solução corrente s de um problema de otimização. Uma pequena alteração nas posições de algumas dessas partículas (perturbação) resulta numa variação de energia, o que equivale a uma variação da função objetivo ao mover-se para uma solução vizinha candidata, isto é, $\Delta = f(s') - f(s)$. Assim temos alguns casos a considerar. Se $\Delta < 0$ (problema de minimização), indica que houve uma redução na função objetivo, o movimento deve ser aceito, a solução vizinha passa a ser a nova solução corrente e o processo continua. Caso $\Delta = 0$, temos um caso de estabilidade, ou equilíbrio, sem alteração de energia, ou seja, $f(s') = f(s)$ já se $\Delta > 0$, temos um aumento de energia que no processo físico é útil para permitir uma futura "acomodação" das partículas, o que significa em outras palavras que é evitada a convergência da função objetivo para mínimos locais indesejáveis,.e a solução vizinha candidata também poderá ser aceita, com uma probabilidade $e^{-\Delta/T}$, onde T é

um parâmetro do método, chamado de temperatura e que regula a probabilidade de se aceitar soluções de pior custo.

Inicialmente a temperatura T atinge um valor elevado T_0 para evitar o bloqueio nas configurações com ótimo local, fazendo com que soluções piores sejam aceitas com uma maior probabilidade. Após um número fixo de iterações, ou seja, até que o sistema atinja o equilíbrio térmico em uma dada temperatura, a temperatura começa a ser reduzida gradualmente por uma razão de resfriamento α, tal que $T_k \leftarrow \alpha \times T_{k-1}$, sendo $0 < \alpha < 1$. Com esse procedimento é possível observar que à medida que T aproxima-se de zero, o algoritmo comporta-se como um método de descida, uma vez que diminui a probabilidade de se aceitar movimentos de piora $\left(T \rightarrow 0 \Rightarrow e^{-\Delta/T} \rightarrow 0\right)$.

Os parâmetros de controle (ou calibragem) do procedimento são: a razão de resfriamento α, o número de iterações para cada temperatura $(SAMax)$ e a temperatura inicial (T_0).

A Figura 3.1 mostra um algoritmo Simulated Annealing básico aplicado a um problema de minimização.

3.3.2 Busca tabu (BT)

Originária dos trabalhos independentes de Fred Glover [16] e Pierre Hansen [19], a metaheurística Busca Tabu (BT) objetiva desenvolver e explorar um conjunto de princípios de buscas inteligentes na solução de problemas, fazendo uso de uma memória flexível [8, 14, 17, 33]. Essa memória inclui processos de criação e exploração de estruturas que consideram a história dos movimentos, fazendo uso de listas tabu para evitar a análise de solução já consideras nas iterações anteriores, permitindo a aceitação de movimentos de piora para tentar escapar de ótimos locais, ainda distantes de um ótimo global.

Podemos dizer que a Busca Tabu é formada basicamente pela construção de uma solução inicial viável, geração de listas tabus e definição do critério de aspiração. Sabe-se que a Busca Tabu é uma variante dos métodos de busca local, que foi uma das primeiras técnicas utilizadas na resolução de problemas de otimização combinatória. Dado um problema de minimização com função objetivo f, começando com uma solução inicial s_0, um algoritmo Busca Tabu, a cada iteração, explora um subconjunto W de $N(s)$, onde $N(s)$ é o conjunto da vizinhança da solução atual s. O melhor valor $s' \in W$, segundo a função $f(\cdot)$, torna-se a nova solução atual mesmo que s' seja pior que s, ou seja, que $f(s') > f(s)$, e o processo continua.

Utiliza-se o critério de melhor vizinho com o objetivo de escapar de um ótimo local. Porém, essa decisão, como dito anteriormente, permite aceitar uma solução pior que a anterior, e isso pode trazer alguns problemas como: a possibilidade de ciclagem, ou seja, o retorno a uma solução já analisada e a repetição de um movimento.

De modo a tentar evitar que estes problemas ocorram, introduz-se uma lista tabu T, ou seja, uma lista de movimentos proibidos (ou tabus). Uma das maneiras de armazenamento na lista tabu, consiste em guardar os movimentos reversos aos últimos $|T|$ movimentos realizados (onde $|T|$ é um parâmetro de entrada) e, funciona como uma fila de tamanho fixo, ou seja, quando adiciona-se um novo movimento à lista, o movimento mais antigo sai [39]. Excluindo-se assim os vizinhos $s' \in W \subset N(s)$ que são obtidos de s por movimentos m que constam na lista tabu.

Por um lado a lista tabu reduz o risco de ciclagem, pois tenta impossibilitar o não retorno, por $|T|$ iterações, a sua solução já visitada, por outro lado, se $|T| >> 0$, armazenar $|T|$ movimentos pode tornar-se inviável na prática, devido a memória necessária para guardar esses movimentos (além

de verificar se $m \in T$ ou se $m \notin T$). Outro problema é que a lista tabu também pode proibir movimentos para soluções que ainda não foram visitadas. Assim, segundo Ochi [33], pode-se deduzir que o mais razoável nos parece limitar o tamanho da lista tabu para que nela sejam armazenados apenas os últimos $|T|$ movimentos realizados. Dessa forma, observamos que o tamanho da lista tabu é um importante parâmetro a ser configurado e pode permanecer fixo ou variável durante a pesquisa.

Ainda segundo Ochi [33], as listas tabu sozinhas nem sempre são um bom filtro, sendo necesários complementá-las com outros mecanismos para que juntos passem a cumprir bem os objetivos de uma lista tabu. Esses complementos são conhecidos como critérios de aspiração.

O critério de aspiração é um mecanismo que sob certas condições retira o estado de tabu de um movimento. Mais precisamente, para cada possível valor v da função objetivo existe um nível de aspiração $A(v)$: uma solução $s' \in V$ pode ser gerada se $f(s') \leq A(f(s))$, mesmo que o movimento m esteja na lista tabu. A função de aspiração A é tal que, para cada valor v da função objetivo, retorna outro valor $A(v)$, que representa o valor que o algoritmo aspira ao chegar de v. Considerando uma função objetivo de valores inteiros, um exemplo simples de aplicação desta idéia é considerar $A(f(s)) = f(s^*) - 1$ onde s^* é a melhor solução encontrada até então. Neste caso, aceita-se um movimento tabu somente se ele conduzir a um vizinho melhor que s^*.

Outras características importantes na busca tabu são as estratégias de intensificação e diversificação, que são incorporadas ao método, respectivamente, pelas funções de memória de curto prazo e de longo prazo.

As estratégias de intensificação têm por objetivo concentrar a pesquisa em determinadas regiões consideradas promissoras, incorporando atributos

das melhores soluções já encontradas durante essa pesquisa e estimular componentes dessas soluções a fazer parte da solução atual.

As estratégias de diversificação, redirecionam a pesquisa para novas regiões ainda não suficientemente exploradas, com o intuito de atingir todo o espaço de soluções possíveis, evitando-se o bloqueio de um determinado ótimo local que equivale a repetições cíclicas indesejadas.

Os parâmetros de controle do procedimento busca tabu são o tamanho da lista tabu $|T|$, a função de aspiração A e o número máximo de iterações sem melhora no valor da melhor solução ($BTMax$). Normalmente utiliza-se dois critérios para parada numa metaheurística busca tabu. Pelo primeiro, quando atinge-se um certo número máximo de iterações sem melhora no valor da melhor solução, e o segundo critério, é quando atinge-se um limite inferior conhecido (ou perto dele) no valor da melhor solução, evitando assim a execução desnecessária do algoritmo quando uma solução ótima é encontrada ou quando uma solução é considerada suficientemente boa.

Os passos do procedimento busca tabu são mostrados na Figura 3.2.

Na Figura 3.2, $f_{\min}$ é o valor mínimo conhecido da função f, informação que em alguns casos está disponível.

3.3.3 Greedy Randomized Adaptative Search Procedure (GRASP)

Um algoritmo GRASP (*Greedy Randomized Adaptative Search Procedure*) é um método iterativo do tipo *multistart*, proposto em [9], que consiste de duas fases: uma fase de construção, na qual uma solução é gerada, elemento a elemento, e de uma fase de busca local, na qual um ótimo na vizinhança da solução construída é pesquisado. A melhor solução encontrada ao longo de todas as iterações GRASP realizadas é retornada como resultado. O pseudo

código descrito pela Figura 3.3 ilustra um procedimento GRASP para um problema de minimização.

A fase de construção GRASP objetiva fornecer soluções viáveis distintas, produzidas por um procedimento de construção, tal como uma heurística construtiva. Essa fase é também um processo iterativo, além de guloso, adaptativo e randômico, o que deu origem ao nome do método, uma vez que nessa fase uma solução viável é construída iterativamente, e um elemento por vez é inserido na solução parcial.

A cada iteração da fase de construção avalia-se apenas elementos que podem ser adicionados à solução, e para fazer a escolha de qual elemento será adicionado ao conjunto solução deve-se montar uma lista com os elementos possíveis de ser adicionados à solução baseada numa função adaptativa gulosa $g : C \rightarrow \mathcal{R}$ que estima o benefício da seleção de cada um desses elementos, construindo assim uma Lista de Candidatos Restrita (LCR). A escolha do elemento a ser adicionado à solução é aleatória dentre os elementos da LCR. Esta técnica de escolha permite que sejam geradas diferentes soluções a cada iteração do GRASP. A escolha adequada da função gulosa também é muito importante, pois os resultados do GRASP sofrem grande influência por esta escolha.

O GRASP é dito adaptativo pois os benefícios associados com a escolha de cada elemento são atualizados a cada iteração durante a fase de construção refletindo as mudanças oriundas da seleção do elemento selecionado na iteração anterior desta fase.

O pseudo código representado pela Figura 3.4, onde $\alpha \in [0, 1]$ é um parâmetro, descreve a fase de construção do GRASP. Observamos que o parâmetro α controla o nível de gulosidade e aleatoriedade da fase de cons-

trução. Um valor $\alpha = 0$ faz gerar soluções puramente gulosas, enquanto que $\alpha = 1$ faz produzir soluções totalmente aleatórias.

É quase sempre benéfico aplicar uma busca local para tentar melhorar cada solução construída. Um algoritmo de busca local trabalha de maneira iterativa para substituir sucessivamente a solução atual pela melhor solução na vizinhança da solução atual e uma nova busca na vizinhança é inicializada. O procedimento termina quando nenhuma solução melhor é encontrada na vizinhança. A Figura 3.5 descreve o pseudo código da fase de busca local com respeito a uma certa vizinhança $N(s)$ para um problema de minimização.

Experimentos computacionais têm mostrado que os métodos de busca local são mais eficientes quando se possui boas soluções iniciais, ou seja, quanto melhor for a solução construída, menor o esforço computacional para na fase de busca local se encontrar uma boa solução, ou até mesmo a solução ótima.

Segundo Resende [37], uma característica especial do GRASP é a facilidade na qual ele pode ser implementado. Poucos parâmetros necessitam ser configurados e portanto, o desenvolvimento do GRASP pode estar focalizado na implementação eficiente de estruturas de dados para assegurar rápidas iterações. Os parâmetros de controle do GRASP são: o tamanho da LRC, o valor de α e o número máximo de iterações ($GRASPMax$).

3.3.4 *Variable Neighborhood Search* (VNS)

O Método de Pesquisa em Vizinhança Variável (do inglês *Variable Neighborhood Search* VNS) é um método heurístico que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizi-nhanças [20, 21]. Ao contrário de outras metaheurísticas baseadas em busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças

gradativamente mais "distantes"da solução corrente (movimento de *shaking*-perturbação) e focaliza a busca em torno de uma nova solução se, e somente se, um movimento de melhora é realizado.

Em geral, heurísticas de busca executam uma sucessão de trocas locais numa solução inicial que melhora iterativamente o valor da função objetivo, até um mínimo local ser encontrado. Isso é, a cada iteração uma solução melhor s' numa vizinhança $N(s)$ da solução atual s é obtida, até que nenhuma melhoria seja encontrada.

Seja $\mathcal{N}_k$, $(k = 1, \ldots, k_{\max})$, um conjunto finito de estruturas de vizinhanças pré-selecionadas e $\mathcal{N}_k(s)$ o conjunto de soluções na $k - ésima$ vizinhança de s. Heurísticas de busca local usam uma única estrutura de vizinhança, isto é, $k_{\max} = 1$.

Na sua forma básica, VNS explora um conjunto de vizinhos da solução atual, faz uma busca local numa solução vizinha a um ótimo local, e move para ele se houver melhoria. Uma busca local básica consiste da aplicação de movimentos de melhoria, enquanto tal movimento exista, e um algoritmo VNS pode ser implementado pela combinação de uma série de movimentos e buscas locais. Esse método oferece diferentes graus de flexibilidade: a ordem na busca local pode ser trocada, a escolha da $\mathcal{N}_k$ a ser usada, e a estratégia de busca escolhida para ser usada nas trocas das vizinhanças.

Na Figura 3.6 temos o pseudo código de um algoritmo VNS. Os parâmetros de controle do VNS são: o número de estruturas de vizinhança e o critério de parada a ser adotado (número máximo de iterações ou o número máximo de iterações sem melhora no valor da melhor solução).

O método *General Variable Neighborhood Search* (GVNS) aplica duas séries de estruturas de vizinhanças diferentes: uma para a perturbação (*shaking*) e outra para a busca local. O GVNS conduz a algumas aplicações com

êxito da metaheurística VNS descrita acima. O procedimento de perturbação aplica um número k de movimentos aleatórios enquanto a busca um número l de movimentos. A busca local pode ser realizada através de um método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND). O pseudo código do método GVNS é apresentado na Figura 3.7.

procedimento SA($f(.)$, $N(.)$, α $SAMax$, T_0, s)

1. $s^* \leftarrow s$;
2. $IterT \leftarrow 0$;
3. $T \leftarrow T_0$;
4. **enquanto** $(T > 0)$ **faça**
5. **enquanto** $(IterT < SAMax)$ **faça**
6. $IterT \leftarrow IterT + 1$;
7. Gere um vizinho qualquer $s' \in N(s)$;
8. $\Delta = f(s') - f(s)$;
9. **se** $(\Delta < 0)$ **então**
10. $s \leftarrow s'$;
11. **se** $(f(s') < f(s^*))$ **então** $s^* \leftarrow s'$;
12. **senão**
13. Tome $x \in [0, 1]$;
14. **se** $(x < e^{-\Delta/T})$ **então** $s^* \leftarrow s'$;
15. **fim-se**;
16. **fim-enquanto**;
17. $T \leftarrow \alpha \times T$;
18. $IterT \leftarrow 0$;
19. **fim-enquanto**;
20. $s \leftarrow s^*$;
21. retorne s ;

fim SA.

Figura 3.1: Algoritmo Simulated Annealing.

procedimento BT($f(.)$, $N(.)$, $A(.)$ $|V|$, f_{min}, $|T|$, $BTMax$, s)

1. $s^* \leftarrow s$;
2. $IterT \leftarrow 0$;
3. $MelhorIter \leftarrow 0$;
4. $T \leftarrow \emptyset$;
5. Inicialize a função de aspiração A;
6. **enquanto** $(f(s) > f_{min})$ e $(IterT - MelhorIter < BTMax)$ **faça**
7. $\quad IterT \leftarrow IterT + 1$;
8. $\quad$ Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subset N(s)$ tal que
   o movimento m não seja tabu $(m \notin T)$ ou s' atenda a condição
   de aspiração $(f(s') < A(f(s)))$;
9. $\quad T \leftarrow T-$ {movimento mais antigo} + {movimento que gerou s'};
10. $\quad$ Atualize a função de aspiração A;
11. $\quad s \leftarrow s'$;
12. $\quad$ **se** $(f(s) < f(s^*))$ **então**
13. $\quad\quad s^* \leftarrow s$;
14. $\quad\quad MelhorIter \leftarrow Iter$;
15. $\quad$ **fim-se**;
16. **fim-enquanto**;
17. $s \leftarrow s^*$;
18. retorne s ;

fim BT.

Figura 3.2: Algoritmo de Busca Tabu.

procedimento GRASP($f(.)$, $g(.)$, $N(.)$ $GRASPMax$, s)

1. $f^* \leftarrow \infty$;
2. **para** $(Iter = 1, 2, \ldots, GRASPMax)$ **faça**
3. Construcao($g(.)$, α, s);
4 Busca Local $(f(.), N(.), s)$;
5. **se** $(f(s) < f^*)$ **então**
6. $s^* \leftarrow s$;
7. $f^* \leftarrow f(s)$;
8. **fim-se**
9. **fim-para**
10. $s \leftarrow s^*$;
11. retorne(s).

fim GRASP.

Figura 3.3: Algoritmo GRASP para minimização.

procedimento `Construcao`($g(\cdot)$, α, s);

1. $s \leftarrow \emptyset$;
2. Inicialize o conjunto C de candidatos;
3. **enquanto** $(C \neq 0)$ **faça**
4. $\quad g(t_{min}) = \min\{g(t) \mid t \in C\}$;
5. $\quad g(t_{max}) = \max\{g(t) \mid t \in C\}$;
6. $\quad LCR = \{t \in C \mid g(t) \leq g(t_{min}) + \alpha(g(t_{max}) - g(t_{min}))\}$;
7. $\quad$ Selecione aleatoriamente, um elemento $t \in LCR$;
8. $\quad s \leftarrow s \cup \{t\}$;
9. $\quad$ Atualize o conjunto C de candidatos;
10. **fim-enquanto**;
11. retorne(s);

fim `Construcao`.

Figura 3.4: A fase de construção de um algoritmo GRASP.

procedimento `BuscaLocal`($f(\cdot)$, $N(\cdot)$, s)

1. $V = \{s' \in N(s) \mid f(s') < f(s)\}$;
2. **enquanto** $|V| > 0$ **faça**
3. Selecione $s' \in V$;
4. $s \leftarrow s'$;
5. $V = \{s' \in N(s) \mid f(s') < f(s)\}$;
6. **fim-enquanto**
7. retorne(s).

fim `BuscaLocal`.

Figura 3.5: A fase de busca local de um algoritmo GRASP.

procedimento VNS($f(\cdot)$, $N(\cdot)$, s_0, r, s)

1. Seja s_0 uma solução inicial;
2. Seja r o número de estruturas diferentes de vizinhanças;
3. $s \leftarrow s_0$; (solução corrente)
4. **enquanto** (a condição de parada não for satisfeita) **faça**
5. $k \leftarrow 1$; (tipo de estrutura de vizinhança corrente)
6. **enquanto** $k \leq r$ **faça**
7. Gere um vizinho qualquer $s' \in \mathcal{N}_k(s)$; (*Shaking*)
8. $s'' \leftarrow BuscaLocal(r')$;
9. **se** $f(s'') < f(s')$
10. **então**
11. $s \leftarrow s''$;
12. $k \leftarrow 1$;
13. **senão**
14. $k \leftarrow k + 1$;
15. **fim-se**
16. **fim-enquanto**
17. **fim-enquanto**
18. retorne(s).

fim VNS.

Figura 3.6: Pseudo-código do VNS para minimização.

procedimento GVNS($f(\cdot)$, $N(\cdot)$, k_{max}, s, $MaxIterGVNS$)

1. // Inicialização
2. Selecione um conjunto de vizinhos N_k, para cada $k = 1, \ldots, k_{max}$;
3. $s^* \leftarrow s$;
4. $Iter \leftarrow 1$;
5. **enquanto** (a condição de parada não for satisfeita) **faça**
6. $k \leftarrow 1$;
7. **enquanto** $k \leq k_{max}$ **faça**
8. // *Shaking*
9. Gere um vizinho qualquer $s' \in \mathcal{N}_k(s)$;
10. // Busca Local
11. Aplique uma busca local em s' até encontrar um mínimo local s'';
12. // Atualização
13. **se** $f(s'') < f(s)$
14. **então**
15. $s \leftarrow s''$;
16. $k \leftarrow 1$;
17. **senão** $k \leftarrow k + 1$;
18. **fim-se**
19. **fim-enquanto**
20. **fim-enquanto**
21. return(s).

fim GVNS.

Figura 3.7: Pseudo-código do GVNS para minimização.

Capítulo 4

Heurísticas básicas propostas para o PCMRC

4.1 Introdução

As metaheurísticas têm a habilidade de escapar de ótimos locais e de encontrar a solução ótima ou próxima dela de uma maneira eficiente. Mas, precisa-se fazer um extensivo ajuste de parâmetros empiricamente, uma vez que a qualidade das soluções é usualmente sensível às suas variações. Assim, encontrar uma heurística adequada ao Problema do Ciclo Mediano sem Restrições de Capacidade requer um esforço substancial. Neste capítulo detalharemos a forma como foi representada a solução, a estrutura de vizinhança e uma descrição de cada uma das heurísticas básicas que foram aplicadas ao nosso problema.

4.2 Representação da solução

Uma solução S para o PCMRC corresponde a um ciclo que possui o depósito v_1, que representa o início da ordenação dos vértices no ciclo, e os demais vértices do grafo que são atribuídos aos vértices do ciclo.

Como a representação da solução é essencial para o sucesso de qualquer método de busca, resolvemos seguir a mesma representação adotada por Pérez e outros em [34]. Assim, uma solução S será representada por um vetor $[v_i : i = 1, ..., n]$ onde v_i representa o $i - ésimo$ elemento da solução. Nesta representação, assumiremos que os m primeiros elementos do vetor pertencem ao ciclo e os $n - m$ elementos restantes não pertencem ao ciclo.

4.3 Função objetivo

Similar ao descrito por Pérez e outros em [34], a função de avaliação (ou objetivo), consiste de minimizar uma combinação convexa dos custos de roteamento e atribuição. Assim, dada uma solução S, o custo de roteamento é dado por:

$$LC(S) := \sum_{i=2}^{m} c_{v_{i-1},v_i} + c_{v_m,v_1}, \tag{4.1}$$

enquanto que o custo de atribuição é dado por:

$$AC(S) := \sum_{i=m+1}^{n} \min_{j=1,...,m} d_{v_i,v_j}. \tag{4.2}$$

A equação (4.1) calcula a soma dos custos de ligação entre os pares de vértices (v_{i-1}, v_i), segundo a ordem no ciclo. O primeiro termo do somatório (v_1, v_2) diz respeito ao custo da ligação do vértice depósito v_1 ao primeiro vértice no ciclo, e o último termo no somatório (v_m, v_1) dá o custo de ligação do último vértice do ciclo até o depósito. Enquanto a equação (4.2) calcula a soma dos

custos de ligação entre os pares de vértices (v_i, v_j), que diz respeito ao custo de ligação de todos os vértices v_i que estão fora do ciclo aos vértices v_j que estão no ciclo.

Uma solução ótima S^* para o Problema do Ciclo Mediano sem Restrições de Capacidade seria uma solução que minimize

$$LC(S^*) + AC(S^*). \tag{4.3}$$

4.4 Estrutura da vizinhança

Como os algoritmos que utilizaremos são baseado na exploração de vizinhanças para resolver o PCMRC, precisamos definir alguns movimentos que gerem soluções vizinhas à partir de uma solução inicial S. Estes movimentos correspondem a uma mo- dificação parcial de uma solução corrente S. O conjunto de soluções obtidas através de um movimento será denotado por $N'(S)$. Para o PCMRC definiremos os seguintes movimentos:

- `adição de vértices no ciclo` (*add*): para inserirmos vértices no ciclo, precisamos saber em qual posição vamos inserí-lo, ou seja, devemos inserir o vértice na posição que proporcione um menor acréscimo no custo de roteamento. O custo de roteamento, obtido pela adição de um novo vértice v_j no ciclo, após o vértice v_k, é dado por:

$$LC(S_{kj}) := LC(S) + d_{v_k,v_j} + d_{v_j,v_{k+1}} - d_{v_k,v_{k+1}}, \tag{4.4}$$

para $k = 1, ..., m-1$, e

$$LC(S_{kj}) := LC(S) + d_{v_m,v_j} + d_{v_j,v_1} - d_{v_m,v_1},$$

para $k = m$.

Assim, o novo vértice a ser inserido no ciclo corresponderá aquele que possuir o menor $LC(S_{kj})$.

O novo custo de atribuição, com a inserção do vértice v_j, é dado por:

$$AC(S_{kj}) := AC(S) - \sum_{i=m+1}^{n} \max\{0, \min_{h=1,...,m} d_{v_i,v_h} - d_{v_i,v_j}\}. \quad (4.5)$$

Numa dada solução o número de vértices no ciclo é igual a m, restando $n-m$ vértices que podem ser inseridos no ciclo, portanto o número máximo de movimentos desta vizinhança é $O(n)$, dado que, em geral temos $3 \leq m \leq \frac{n}{2}$.

- `remoção de vértices no ciclo` (*drop*): consideremos S_k a solução obtida após a remoção do $k-ésimo$ vértice do ciclo, onde $k = 2, ..., m$. Assim, o custo de roteamento ao se remover um vértice v_k do ciclo S e conectar o vértice v_{k-1} diretamente ao vértice v_{k+1}, é expresso por:

$$LC(S_k) := LC(S) + d_{v_{k-1},v_{k+1}} - d_{v_{k-1},v_k} - d_{v_k,v_{k+1}}, \quad (4.6)$$

para $k = 2, ..., m-1$, e

$$LC(S_k) := LC(S) + d_{v_{m-1},v_1} - d_{v_{m-1},v_m} - d_{v_m,v_1}, \quad (4.7)$$

para $k = m$.

Já o novo custo de atribuição deve ser recalculado para todos os vértices próximos ao vértice excluído, reatribuindo os vértices previamente atribuídos a v_k, conectando-os agora ao vértice mais próximo no ciclo e também atribuindo o vértice v_k a um vértice no ciclo. O vértice v_k a ser removido é escolhido de modo a gerar um menor aumento na função objetivo 4.3.

Numa dada solução pode-se retirar algum dos m vértices do ciclo, até que o limite de $m \geq 3$ seja alcançado, portanto o número máximo de movimentos desta vizinhança é $O(m)$.

- `troca de dois vértices` (*swap*): esse movimento consiste de uma combinação de um movimento de adição e um movimento de remoção de vértices, descritos anteriormente. Um vértice é removido do ciclo e outro é inserido na melhor posição possível, ou seja, numa posição que propicie uma diminuição nos custos de roteamento e atribuição. Os custos de roteamento e atribuição são calculados como segue. Seja S_{ij}, $1 < i \leq m$ e $m < j \leq n$, a nova solução consistindo na troca do vértice v_i do ciclo e v_j fora do ciclo, adicionando v_j na melhor posição possível, logo após v_k. Com isto, o custo de roteamento $LC(S_i)_{kj}$ pode ser obtido pela fórmula 4.4 para adição e remoção de um vértice. O novo custo de atribuição será obtido por:

$$AC(S_{ij}) := \min \left\{ d_{v_i,v_j}, \min_{\substack{l=1,\ldots,m \\ l \neq i}} d_{v_i,v_l} \right\} + \sum_{k=m+1}^{n} \min\{d_{v_k,v_j}, \min_{\substack{l=1,\ldots,m \\ l \neq i}} d_{v_k,v_l}.\} \tag{4.8}$$

  Dada uma solução pode-se trocar um dos m vértices do ciclo por qualquer um dos $n-m$ vértices que estão fora do ciclo, portanto o número máximo de movimentos desta vizinhança é $O(nm)$.

- `troca de duas arestas` (*2-opt*): Seja C o ciclo da solução S. Substitua duas arestas não-adjacentes do ciclo por outras duas arestas, de tal maneira, que o novo custo do ciclo seja minimizado. Continue até que nenhuma melhoria seja mais possível.

  O número máximo de movimentos desta vizinhança é $O(m^2)$.

- `troca de três arestas` (*3-opt*): O procedimento 3-*opt* é similar ao 2-*opt* trocando-se três arestas não-adjacentes do ciclo por outras três arestas, de modo que se obtenha um mínimo custo, até que nenhuma modificação sem melhoria possa ser efetuada.

O número máximo de movimentos desta vizinhança é $O(m^3)$.

Esses dois últimos procedimentos são casos especiais do procedimento de *k-opt* onde trocas entre k arestas são feitas, de tal maneira que, um ciclo com um custo de roteamento menor seja encontrado.

4.5 Construção de uma solução inicial

Uma solução inicial para o PCMRC pode ser obtida através de uma heurística construtiva. Estas heurísticas construtivas consistem de uma série de movimentos de adição de vértices no ciclo parcial. Durante o processo de adição dos vértices à solução corrente, o custo de roteamento aumenta enquanto que o custo de atribuição diminui. O procedimento pára quando uma solução obtida é viável e não é possível melhorar a solução atual.

Alguns exemplos de heurísticas construtivas que podem ser empregadas em problemas que possuem um ciclo como solução são:

- `método de inserção aleatória`: um novo elemento é escolhido aleatoriamente para ser adicionado ao ciclo atual;
- `método da melhor inserção`: um elemento é selecionado de modo a minimizar o incremento no custo de roteamento da solução;
- `método de inserção mais barata`: um novo elemento é selecionado de modo a maximizar a diminuição no custo de atribuição da nova solução;

A solução inicial do PCMRC é gerada de forma construtiva e usando a seguinte técnica. Para construirmos a solução inicial iniciamos apenas com o

depósito e inserimos mais dois vértices com melhor média de conectividade dada para cada v_j por:

$$atrib_{med}(v_j) = \frac{\sum_{i=2}^{m} d_{v_j,v_i}}{m}$$

em relação aos demais vértices.

Até este ponto temos uma solução viável para o PCMRC, cujo ciclo solução possui três vértices. Entretanto, esta solução pode ser ou não melhorada. O que irá determinar isso será a adição de novos vértices ao ciclo C da solução.

Para determinar o próximo vértice a ser adicionado à solução atual usa-se a média de conectividade dos vértices do ciclo. Mais, formalmente, para cada $v_i \in V \backslash S$, temos:

$$MenorIncremento(v_i) + \frac{\sum_{j \in V \backslash S} d_{v_i,v_j}}{|V \setminus S|}. \tag{4.9}$$

A escolha do novo vértice a ser inserido no ciclo corresponde aquele com menor valor na equação (4.9).

4.6 Algoritmos propostos

4.6.1 GRASP

Como foi dito na subseção 3.3.3 do capítulo anterior o GRASP é um processo *multistart* e iterativo, onde diferentes pontos no espaço de busca são testados com busca local [9, 36]. Nesta seção apresentaremos o GRASP proposto para a resolução do PCMRC. Primeiro, será descrito a fase de construção. Em seguida, a fase de busca local.

Fase de construção

A fase de construção é iterativa, adaptativa, gulosa e aleatória, no sentido que cada elemento escolhido em cada iteração durante a fase de construção é dependente daquele escolhido previamente, como visto na subseção 3.3.3.

procedimento `ConstroiSolucaoGulosaAleatoria(MaxIterFilter, `$g(\cdot)$`, `α`)`

1. $f(S^*) \leftarrow \infty$;
2. **para** `IterFilter` $\leftarrow 1$ até `MaxIterFilter` **faça**
3. $S \leftarrow v_1$;
4. **enquanto** a condição de parada não for satisfeita faça
5. $g_{min} = \min\{g(v_i) \mid v_i \in V \setminus S\}$;
6. $g_{max} = \max\{g(v_i) \mid v_i \in V \setminus S\}$;
7. $L = \{v_i \in V \setminus S \mid g(v_i) \leq g_{min} + \alpha(g_{max} - g_{min})\}$;
8. Selecione aleatoriamente um vértice $v_i \in L$;
9. $S = S \cup \{v_i\}$;
10. $V = V \setminus \{v_i\}$;
11. **fim-enquanto**
12. **se** $(f(S) < f(S^*))$ **então**
13. $S^* \leftarrow S$;
14. $f(S^*) \leftarrow f(S)$;
15. **fim-se**
16. **fim-para**
17. retorne(S^*).

fim `ConstroiSolucaoGulosaAleatoria`.

Figura 4.1: Pseudo-código para a fase de construção GRASP.

Em seguida, descreveremos uma função adaptativa gulosa $g : L \rightarrow \mathcal{R}$, um mecanismo de construção para a LCR, e um critério de seleção probabilístico para a fase de construção proposta.

Para obtermos um ciclo inicial começamos apenas com o vértice depósito v_1 e mais dois outros vértices conforme descrito na seção 4.5. A função gulosa é definida de acordo com a fórmula 4.9, ou seja,

$$g(v_i) = MenorIncremento(v_i) + \frac{\sum_{j \in V \setminus S} d_{v_i,v_j}}{|V \setminus S|}. \quad (4.10)$$

A escolha gulosa é selecionar um vértice v_i com menor $g(v_i)$.

A componente probabilística do GRASP é caracterizada pela escolha aleatória de um dos melhores candidatos da lista, mas não necessariamente o melhor candidato. A lista dos melhores candidatos corresponde a Lista de Candidatos Restrita (LCR). Para definir o mecanismo de construção proposto para a LCR, considere os seguintes valores:

$$g_{\min} = \min\{g(v_i) \mid v_i \in V \setminus S\} \text{ e } g_{\max} = \max\{g(v_i) \mid v_i \in V \setminus S\} \quad (4.11)$$

Para selecionar um vértice a ser adicionado à solução, uma LCR é definida para incluir todos os vértices v_i no conjunto de candidatos L tendo custo $g(v_i) \leq g_{\min} + \alpha(g_{\max} - g_{\min})$ onde $\alpha \in [0,1]$ corresponde ao fator de gulosidade. Em seguida, um vértice $v_i \in L$ é escolhido aleatoriamente, e é então adicionado à solução, ou seja, $S = S \cup \{v_i\}$.

Uma outra característica adotada na heuristica é o uso de um filtro na fase de construção. Ao contrário do GRASP padrão, onde a cada iteração uma única solução inicial é gerada e em seguida uma busca local; no GRASP com filtro o algoritmo de construção é executada $MaxIterFilter$ vezes, construindo-se assim $MaxIterFilter$ soluções iniciais diversificadas. Dessas $MaxIterFilter$ soluções, selecionaremos somente a melhor delas para a etapa de busca local. Assim, espera-se que com uma solução inicial de melhor qualidade, a solução final obtida após a busca local, também poderá ser melhor.

A Figura 4.1 apresenta o pseudo-código correspondente a esta fase.

Fase de busca local

É sempre benéfico aplicar uma busca local para tentar melhorar cada solução construída numa heurística GRASP já que a solução gerada na fase de construção não representa necessariamente o ótimo local.

O algoritmo de busca local proposto trabalha de uma forma iterativa substituindo sucessivamente a solução atual por uma solução melhor na vizinhança da solução atual e uma nova busca na vizinhança é inicializada. A busca termina quando um ótimo local de boa qualidade é alcançado.

Faremos uso dos movimentos descritos na seção 4.4 para a definição das vizinhanças da fase de busca local proposta para o PCMRC. Assim, dada uma solução S, cada movimento é aplicado a S, obtendo-se um conjunto com todas as soluções vizinhas melhores que a solução S. Desta lista, seleciona-se o melhor vizinho, ou seja, aquele que minimiza a soma dos custos de roteamento e de atribuição como definida em 4.1 e 4.2. A busca local termina quando nenhuma melhoria adicional é possível.

A figura 4.2 mostra o pseudo-código da fase de busca local proposta para o PCMRC.

4.6.2 Busca tabu

Para empregar busca tabu ao Problema do Ciclo Mediano sem Restrições de Capacidade utiliza-se de um procedimento de pesquisa local juntamente com um procedimento de inicialização que procura gerar boas soluções iniciais com o objetivo de diminuir o tempo de pesquisa do algoritmo. A solução inicial foi gerada como descrito na seção 4.5.

A partir dessa solução inicial, a metaheurística busca tabu segue, explorando toda a vizinhança $N(S)$ da solução atual S, iterativamente, e de

procedimento `BuscaLocal`$(f(\cdot), N(\cdot), s)$

1. $H_{add} \leftarrow \{s' \in N_{add}(s) \mid f(s') < f(s)\}$;
2. $H_{drop} \leftarrow \{s' \in N_{drop}(s) \mid f(s') < f(s)\}$;
3. $H_{add/drop} \leftarrow \{s' \in N_{add/drop}(s) \mid f(s') < f(s)\}$;
4. $H_{2-opt} \leftarrow \{s' \in N_{2-opt}(s) \mid f(s') < f(s)\}$;
5. $H \leftarrow H_{add} \cup H_{drop} \cup H_{add/drop} \cup H_{2-opt}$;
6. **enquanto** $|H| > 0$ **faça**
7. Selecione $s' \in H$;
8. $s \leftarrow s'$;
9. **fim-enquanto**
10. retorne(s).

fim `BuscaLocal`.

Figura 4.2: Pseudo-código para a fase de busca local GRASP.

maneira determinística, por meio dos movimentos de adição, remoção, troca de vértices, bem como 2-opt, conforme definidos na seção 4.4. e guiados pela função objetivo 4.3. O algoritmo segue, então, para o melhor vizinho S' segundo a função objetivo, e este novo vizinho torna-se a nova solução atual $f(S')$, mesmo que S' seja pior que S, ou seja, que $f(S') > f(S)$ e o processo continua.

Como mencionado na seção 3.3.2 existe a possibilidade de ciclagem do algoritmo e, de modo a reduzir este risco, introduziu-se uma lista tabu T, que armazena os movimentos realizados. Os elementos são armazenados segundo um critério FIFO onde o primeiro elemento inserido é o primeiro a ser retirado, ou seja, os elementos vão sendo colocados na lista e retirados (ou processados) por ordem de chegada. A lista tabu de tamanho fixo contém os

$|T|$ movimentos mais recentes e reduz o risco de se revisitar uma das $|T|-1$ soluções anteriormente visitadas.

Como a lista tabu sozinha nem sempre é um bom filtro [33] o algoritmo BT proposto usa também uma função de aspiração $A(f(s))$, onde $A(f(s')) = f(s^*)-1$ sendo s^* é a melhor solução obtida para o PCMRC até então. Dessa forma, um dos movimentos definidos para o PCMRC perde seu estado tabu se uma solução s' gerada satisfizer a seguinte condição $f(s') \leq f(s^*)$.

Os parâmetros de controle do procedimento busca tabu são o tamanho da lista tabu $|T|$, a função de aspiração e o número máximo de iterações sem melhora no valor da melhor solução ($BTMax$). O procedimento pára quando atinge o $BTMax$.

Os passos para o procedimento busca tabu para o Problema do Ciclo Mediano sem Restrições de Capacidade são mostrados na figura 4.3

4.6.3 *Simulated anealling*

O algoritmo descrito na figura 4.4 faz uso da solução inicial gerada pelo procedimento descrito na seção 4.5. A temperatura inicial (T_0) é fixada num valor elevado para evitar que soluções piores sejam aceitas com uma maior probabilidade. Para um número fixo de iterações para cada temperatura ($SAMax$), escolhemos aleatoriamente um dos movimentos de adição, remoção, troca de vértices ou 2-opt descrito na seção 4.4, para gerarmos o melhor vizinho da solução atual de acordo com o movimento escolhido.

Os parâmetros de controle são a razão de resfriamento α, o número de iterações para cada temperatura e a temperatura inicial. O procedimento pára quando atingir o número máximo de iterações.

procedimento BT($f(.)$, $N(.)$, $A(.)$ $|V|$, f_{min}, $|T|$, $BTMax$, s)

1. $s^* \leftarrow s$;
2. $IterT \leftarrow 0$;
3. $MelhorIter \leftarrow 0$;
4. $T \leftarrow \emptyset$;
5. Inicialize a função de aspiração A;
6. **enquanto** $(IterT - MelhorIter < BTMax)$ **faça**
7. $\quad IterT \leftarrow IterT + 1$;
8. $\quad$ Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subset N(s)$, tal que,
   o movimento m não seja tabu $(m \notin T)$ ou s' atenda a condição
   de aspiração $f(s') < A(f(s))$;
9. $\quad T \leftarrow T-$ {movimento mais antigo} + {movimento que gerou s'};
10. $\quad$ Atualize a função de aspiração A;
11. $\quad s \leftarrow s'$;
12. $\quad$ **se** $(f(s) < f(s^*))$ **então** $s^* \leftarrow s$;
13. $\quad\quad MelhorIter \leftarrow Iter$;
14. $\quad$ **fim-se**;
15. **fim-enquanto**;
16. retorne (s^*) ;

fim BT

Figura 4.3: Pseudo-código para a Busca Tabu.

procedimento `EscolheMovAleatorio` ()

1. $itAtual \leftarrow rand()$;
2. **avalie** ($itAtual\ MOD\ 4$)
3. caso 0: $s' \leftarrow add()$;
4. caso 1: $s' \leftarrow drop()$;
5. caso 2: $s' \leftarrow swap()$;
6. caso 3: $s' \leftarrow 2 - opt()$;
7. **fim**-avalie
8. retorne (s')

fim `EscolheMovAleatorio`.

Figura 4.4: Algoritmo para escolher aleatoriamente um movimento.

procedimento SA($f(.)$, $N(.)$, α $SAMax$, T_0, s)

1. $s^* \leftarrow s$;
2. $IterT \leftarrow 0$;
3. $T \leftarrow T_0$;
4. **enquanto** $(T > 0)$ **faça**
5. **enquanto** $(IterT < SAMax)$ **faça**
6. $IterT \leftarrow IterT + 1$;
7. EscolheMovAleatorio();
8. $\Delta = f(s') - f(s)$;
9. **se** $(\Delta < 0)$ **então**
10. $s \leftarrow s'$;
11. **se** $(f(s') < f(s^*))$ **então** $s^* \leftarrow s'$;
12. **senão**
13. Tome $x \in [0, 1]$;
14. **se** $(x < e^{-\Delta/T})$ **então** $s^* \leftarrow s'$;
15. **fim-se**;
16. **fim-enquanto**;
17. $T \leftarrow \alpha \times T$;
18. $IterT \leftarrow 0$;
19. **fim-enquanto**;
20. retorne (s^*) ;

fim SA.

Figura 4.5: Pseudo-código para o Simulated Annealing.

Capítulo 5

Metaheurísticas híbridas propostas para o PCMRC

Devido ao fraco desempenho computacional das heurísticas básicas na resolução do Problema do Ciclo Mediano sem Restrições de Capacidade, decidimos propor algumas metaheurísticas híbridas para o PCMRC. Inicialmente, descreveremos na seção 5.1 uma heurística Busca Tabu que faz uso de uma lista tabu dinâmica e de uma estratégia de oscilação. Esta heurística irá trabalhar conjuntamente com uma heurística *Simulated Annealing* e um GRASP. Estas metaheurísticas híbridas estão descritas nas seções 5.2 e 5.3. Por último, na seção 5.4, descreveremos a metaheurística híbrida definida a partir de um GRASP e de um método de Pesquisa de Vizinhança Variável.

5.1 Busca Tabu

5.1.1 Listas tabu dinâmica

Nós usamos a mesma lista tabu, definida na seção 4.7, a fim de evitar a visita, por um determinado período, a soluções já visitadas. Entretanto, o comprimento desta lista varia durante a busca entre 1/2 e 3/2 de um valor dado do *tabu-tenure* l_i. Assim, podemos diminuir o valor de l_i para intensificar a busca dentro das regiões promissoras, enquanto que podemos aumentar este valor para que possamos sair das regiões não promissoras. Para facilitar o ajuste deste valor definimos uma fase de melhoria por um conjunto de Δip iterações consecutivas que diminuem a função objetivo. Do contrário, uma fase de piora é um conjunto de Δwp iterações consecutivas, tais que, o valor da função objetivo não melhora. O valor inicial de l_i é igual a $|T|$ e varia como segue:

$$l_i = max(l_i - 1, 1/2tabu - tenure_i); i = 1, 2. \tag{5.1}$$

após uma fase de melhoria e,

$$l_i = min(l_i + 1, 3/2tabu - tenure_i); i = 1, 2. \tag{5.2}$$

após uma fase de piora

5.1.2 Estratégia de Oscilação

Como mencionado anteriormente, os experimentos computacionais mostraram que a heurística Busca Tabu básica proposta não se mostrou muito eficiente na resolução do PCMRC. Assim, além do emprego do conceito de listas tabu dinâmicas, procuramos melhorar o desempenho desta técnica us-

ando uma estratégia de vizinhança variável padrão. Esta estratégia consiste numa vi-zinhança adicional e uma regra de troca para substituir a vizinhança atual por uma outra vizinhança.

Assuma que a estrutura de vizinhança atual, representada por N_1, seja formada por r movimentos *add e swap* seguidos por 2-*opt*. Já a nova vizinhança, denotada por N_2, seja constituída por r movimentos *drop*, seguidos por 2-*opt*. Em ambos, o número de movimentos r é escolhido aleatoriamente entre 2, 3, 4 ou 5. Assim, podemos estabelecer que a troca de N_1 por N_2 é controlada pela seguinte condição C_1: um número i_{ni} de iterações consecutivas executadas sem melhora.

A cada iteração nós analisamos a condição C_1 e, se necessário, transformamos a solução atual numa nova através da vizinhança N_2.

Um pseudo código possível de nossa BT com lista dinâmica e estratégia de oscilação é apresentado na Figura 5.1

5.2 Algoritmos propostos

5.2.1 Algoritmo SABT

Partindo de uma solução inicial, obtida pelo procedimento SA descrito na seção 4.6.3, a heurística Busca Tabu proposta segue, iterativamente, explorando toda a vizinhança N_1 ou N_2 da solução atual, através dos movimentos definidos na seção 4.4 e guiados pela função objetivo expressa em 4.3. Vale ressaltar que a lista tabu é dinâmica e empregada conforme seção 5.1.1.

O algoritmo SABT tem como parâmetros o número $MaxIter$ de iterações, a função objetivo, as estruturas de vizinhanças N_1 e N_2, o número de iterações para cada temperatura $SAMax$, a razão de resfriamento, a temperatura inicial, o número máximo de iterações sem alteração no valor da melhor

solução ($BTMax$), o tamanho inicial da lista tabu $|T|$, a função de aspiração, os valores de Δwp e Δip.

5.2.2 Algoritmo GBT

O algoritmo GBT é um procedimento GRASP que faz uso de um algoritmo Busca Tabu para o refinamento de uma solução. Neste algoritmo híbrido, a solução inicial é gerada pelo procedimento GRASP (conforme descrito na seção 4.6.1) e o refinamento desta solução é feito através do método Busca Tabu (seção 5.1, figura 5.1).

A heurística toma como parâmetros o número $MaxIter$ de iterações, o valor *RandomSeed*, usado como uma semente inicial para o gerador de números pseudo-aleatórios, a função objetivo, as estruturas de vizinhanças N_1 e N_2, o número máximo de iterações sem alteração no valor da melhor solução ($BTMax$), o tamanho inicial da lista tabu $|T|$, a função de aspiração, os valores de Δwp e Δip.

5.2.3 Algoritmo GGVNS

O algoritmo GGVNS é um algoritmo baseado em um procedimento GRASP e um algoritmo GVNS. Este último é utilizado para o refinamento das soluções obtidas pelo GRASP.

Em seguida, descrevemos os algoritmos GVNS e GGVNS propostos para o PCMRC.

Algoritmo GVNS

Inicialmente descrevemos em detalhes o GVNS básico proposto para o PCMRC, a fim de facilitar a discussão da aproximação híbrida que seguirá nessa seção.

O algoritmo GVNS proposto para o PCMRC usa um procedimento de perturbação muito simples. Dado um tamanho t para o procedimento de perturbação, escolhemos t vezes, aleatoriamente, dois vértices v_i e v_j. Se o vértice v_i está no ciclo e v_j está fora do ciclo realizamos o movimento de *swap* (conforme seção 4.4), se ambos os vértices estão no ciclo executamos um movimento de remoção do vértice v_i (conforme seção 4.5), e se ambos os vértices estão fora do ciclo adicionamos v_j (conforme seção 4.4).

Além disso, os movimentos de *add, drop, swap*, 2-*opt* e 3-*opt* definidos na seção 4.4 são usados no GVNS básico com vizinhanças N_k $(k = 1, ..., k_{\max})$. Assim, o laço nas linhas $5-18$ investiga uma vizinhança escolhida aleatoriamente, de cada vez, até que um ótimo local com respeito as vizinhanças *add, drop, swap*, 2-*opt* e 3-*opt* encontradas.

Algoritmo GGVNS

Uma maneira natural de usar a hibridização GRASP e GVNS é usar este método na segunda fase ou em uma fase após o método de busca local do GRASP [22]. Nessa seção, descrevemos a nova heurística GRASP combinada com o GVNS [7] para o PCMRC usando o segundo modo.

O pseudo-código na Figura 5.3 ilustra o algoritmo baseado nos algoritmos GRASP e GVNS.

A heurística toma como parâmetros o número $MaxIter$ de iterações, o valor *RandomSeed* usado como uma semente inicial para o gerador de números pseudo-aleatórios, a função objetivo, as estruturas de vizinhanças e o número $MaxIterGVNS$ de iterações do GVNS. O laço nas linhas $2-10$ executa $MaxIter$ iterações. Nas linhas $3-8$ é usado o algoritmo GRASP descrito na seção 4.6. A melhor solução encontrada é atualizada nas linhas $5-8$ a cada iteração e retornada na linha 11. Na linha 8 aplicamos t vezes um movimento escolhido aleatoriamente (conforme seção 4.4 e Figura 4.4). Na

linha 10 é aplicado um método de descida para refinamento da solução obtida na fase de diversificação (*shake*). A estratégia GVNS, usando as vizinhanças *add, drop, swap*, 2-*opt* e 3-*opt*, é implementada na linha 9, como descrito acima.

procedimento BT($f(.)$, $N(.)$, $A(.)$ $|V|$, $|T|$, $BTMax$, s)

1. $s^* \leftarrow s$;
2. $IterT \leftarrow 0$; $MelhorIter \leftarrow 0$;
3. $T \leftarrow \emptyset$;
4. Inicialize a função de aspiração A;
5. **enquanto** $(IterT - MelhorIter < BTMax)$ **faça**
6. $IterT \leftarrow IterT + 1$;
7. Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subset N(s)$ tal que o movimento m não seja tabu ($m \notin T$) ou s' atenda a condição de aspiração ($f(s') < A(f(s))$);
8. Atualize a função de aspiração A;
9. **se** $(f(s') < f(s^*))$ **então** $s^* \leftarrow s'$;
10. $MelhorIter \leftarrow Iter$;
11. **fim-se**;
12. **se** $(f(s') < f(s))$ **então**
13. $ip \leftarrow ip + 1$; $wp \leftarrow 0$;
14. **senão**
15. $wp \leftarrow wp + 1$; $ip \leftarrow 0$;
16. **fim-se**;
17. AtualizaListaTabu(m, ip, wp);
18. **fim-enquanto**;
19. $s \leftarrow s^*$;
20. retorne s ;

fim BT.

Figura 5.1: Algoritmo de Busca Tabu com estratégia de oscilação e lista tabu dinâmica.

procedimento GVNS($f(\cdot)$, $N(\cdot)$, k_{max}, s, MaxIterGVNS)

1. // Inicializacao
2. Selecione um conjunto de vizinhos N_k, para cada $k = 1, \ldots, k_{max}$;
3. $s^* \leftarrow s$;
4. $Iter \leftarrow 1$;
5. **enquanto** $Iter <$ MaxIterGVNS **faça**
6. // Perturbacao
7. $t \leftarrow random(2, 5)$;
8. Aplique t vezes um movimento escolhido, aleatoriamente, a solucao s para obter s';
9. // Busca Local
10. Aplique a busca local para s' ate um minimo local s'' ser encontrado;
11. // Atualize
12. **se** $f(s'') < f(s^*)$
13. **então**
14. $s^* \leftarrow s''$;
15. $s \leftarrow s''$.
16. **fim-se**
17. $Iter \leftarrow Iter + 1$;
18. **fim-enquanto**
19. retorne(s^*).

fim GVNS.

Figura 5.2: Pseudo-código GVNS básico para minimização do PCMRC.

procedimento GGVNS($MaxIter$, $RandomSeed$, $f(\cdot)$, $g(\cdot)$, $N(\cdot)$, k_{max}, α)

1. $f(s^*) \leftarrow \infty$;
2. **para** $i \leftarrow 1$ to MaxIter **faça**
3. $s \leftarrow$ ConstroiSolucaoGulosaAleatoria($MaxIterFilter$, $g(\cdot)$, α);
4. $s' \leftarrow$ BuscaLocal(s);
5. **se** $(f(s') < f(s^*))$ **então**
6. $s^* \leftarrow s'$;
7. $f(s^*) \leftarrow f(s')$;
8. **fim-se**
9. GVNS(f, N, k_{max}, s', $MaxIterGVNS$);
10. **fim-para**
11. retorne(s^*).

fim GGVNS.

Figura 5.3: Pseudo-código da heurística GGVNS.

Capítulo 6

Implementação e resultados computacionais

Neste capítulo apresentaremos alguns resultados computacionais obtidos nos testes realizados durante este trabalho. Na seção 6.1, faremos uma análise comparativa entre os resultados obtidos para cada uma das metaheurísticas híbridas propostas. Na seção 6.2, realizaremos uma comparação entre a melhor das heurísticas híbridas e a heurística proposta em [34].

Os algoritmos propostos nos capítulos 4 e 5 deste trabalho foram implementados em linguagem C++, e todos os experimentos computacionais foram executados num computador SEMPRON 2.6 GHz AMD com 512 MB de memória RAM, operando no sistema Linux Red Hat 9.

6.1 Instâncias do problema

Executamos os testes usando duas classes de instâncias. A primeira classe, denominada C1, corresponde as instâncias aleatórias definidas em [27, 28]. A segunda classe, denominada C2, corresponde a instâncias do Problema do

Caixeiro Viajante. Para cada instância executamos os algoritmos propostos cinco vezes, ou seja, fizemos uso de cinco sementes diferentes.

6.1.1 Classe C1

Testamos as instâncias de classe C1 envolvendo $10, 20, 30, 40, 50, 75$ e 100 vértices, tomando o formato **EUC2D** (distâncias Euclidianas) no cálculo das distâncias. Para definir os custos de roteamento e os custos de atribuição, utilizamos a mesma técnica empregada em [27, 28], ou seja, fazemos $c_{ij} = \beta \times l_{ij}$ e $d_{ij} = (10 - \beta) \times l_{ij}$, onde $\beta \in \{5, 7, 9\}$ e l_{ij} corresponde a distância Euclidiana entre os vértices v_i e v_j.

6.1.2 Classe C2

As instâncias de classe C2 foram testadas envolvendo $51, 52, 76, 99$ e 100 vértices, tomando o formato **EUC2D** (distâncias Euclidianas) no cálculo das distâncias.

6.2 Comparação entre os algoritmos híbridos

Nesta seção ilustraremos uma comparação do uso das metaheurísticas híbridas SABT, GBT e GGVNS.

Para a metaheurística híbrida SABT, os parâmetros utilizados na fase SA foram: a temperatura inicial $T_0 = 1000$, o número máximo de iterações para cada temperatura fixado em $SAMax = 100$, que quando atingido pára o procedimento e os valores da razão de resfriamento pertencente ao conjunto $\{0.80, 0.85, 0.90\}$. Já na fase de busca tabu os parâmetros foram: o número máximo de iterações sem alteração no valor da melhor solução foi fixado em $BTMax = 100$, o tamanho inicial da lista tabu $|T| \in \{100, 150\}$. Após os

testes computacionais, verificamos que o valor apropriado para a taxa de resfriamento foi igual a $0,90$,

Na metaheurística GBT, os parâmetros empregados na fase GRASP foram: o número máximo de iterações foi fixado em $MaxIter = \min(n, 100)$, $\alpha \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Os parâmetros da Busca Tabu foram os mesmos empregados na metaheurística SABT. O melhor valor de α encontrado após os testes computacionais realizados para a metaheurística GBT foi o $\alpha = 0,8$.

A última metaheurística híbrida testada, foi a GGVNS. Os parâmetros utilizados na fase GRASP foram: número máximo de iterações $MaxIter = \min(n, 100)$, número máximo de iterações do filtro $MaxFilter = 25$ e $\alpha \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Já para a fase GVNS, os parâmetros utilizados foram: número de iterações fixado em $MaxIterGVNS = 10$ e $k = 2$. Nesta metaheurística, os melhores valores foram encontrados para $\alpha = 0,8$.

Os resultados dos testes estão apresentados nas tabelas 6.1 a 6.7 em cinco colunas. Na primeira coluna temos o nome das instâncias, que nos traz a informação sobre o número de vértices. A segunda coluna indica os valores para cada β. As três colunas seguintes trazem as informações sobre as metaheurísticas híbridas propostas, apresentando o valor mínimo da função objetivo e o tempo mínimo de execução nas cinco rodadas, para cada uma das metaheurísticas. Como podemos observar nas tabelas, nas instâncias consideradas pequenas, ou seja, com até 30 vértices, temos um equilíbrio quanto à qualidade das soluções, e o tempo computacional é praticamente o mesmo. Já nas instâncias com mais de 30 vértices podemos notar um melhor desempenho da metaheurística GGVNS quanto à qualidade das soluções em relação às demais metaheurísticas propostas SABT e GBT à medida que o valor de n cresce.

Na tabela 6.8 apresentamos um quadro comparativo entre o desempenho das metaheurísticas em relação a solução obtida. Observando esta tabela nota-se que a metaheurística GGVNS manteve-se estável num número baixo de derrotas conservando uma média de 2 a 3 derrotas a cada 30 instâncias testadas. enquanto que a SABT e a GBT para as instâncias com 75 e 100 vértices obtiveram uma média de 25 derrotas a cada 30 instâncias testadas. Assim, de uma maneira geral observamos que para as 210 instâncias testadas, a GGVNS obteve apenas 18 derrotas, enquanto que a GBT obteve 59 e a SABT 74, dessa maneira concluímos que a GGVNS apresentou o menor número de derrotas quando comparada com as outras duas. Podemos verificar ainda que a segunda melhor metaheurística é a GBT. Logo, podemos concluir que a metaheurística GRASP associada a uma metaheurística que amplia a busca na vizinhança consegue resultados melhores para o PCMRC.

6.3 Comparação entre GGVNS e o VNTS [[34]]

Nessa subseção poderemos observar a eficiência da hibridização GRASP + GVNS para resolver o problema do ciclo mediano sem restrições de capacidade. Nestes experimentos foram utilizadas instâncias de classe C1 e C2. Cada uma das 243 instâncias foi executada 5 vezes com diferentes sementes.

Resultados da classe C1

As tabelas 6.9 a 6.15 a seguir, apresentam os resultados obtidos para as 210 instâncias de classe C1 comparativamente para as metaheurísticas GGVNS proposta no capítulo 5 desse trabalho e a metaheurística VNTS

proposta em [34]. Na primeira coluna das tabelas apresentamos o nome das instâncias, na coluna 2 os valores de β correspondentes, nas terceira e quarta colunas apresentamos os valores mínimos da função objetivo 4.3, e o tempo médio de execução de cada uma das metaheurísticas GGVNS e VNTS, e na última coluna temos o desvio padrão (GAP) com relação a melhor solução encontrada em [34]. O GAP é calculado como segue:

$$\frac{f\_min_{(VNTS)} - f\_min_{(GGVNS)}}{f\_min_{(GGVNS)}}. \tag{6.1}$$

Analisando as tabelas verificamos que obtemos melhores resultados em 43 das 210 instâncias testadas para $\alpha = 0,8$. Estes resultados são novos, porém como não temos conhecimento do valor ótimo, não podemos quantificar a qualidade dessas novas soluções. Para as intâncias com até 30 vértices podemos verificar uma competitividade nos algoritmos tanto na qualidade das soluções quanto no tempo computacional, podemos observar por exemplo que nessas 30 instâncias melhoramos os resultados obtidos pela técnica VNTS em 5 instâncias e alcançamos os mesmo resultados em 23 instâncias. Porém para as instâncias maiores, com mais de 40 vértices, observamos uma melhora significativa na qualidade das soluções da GGVNS em relação a VNTS, como por exemplo nas instâncias com 100 vértices onde conseguimos melhorar os resultados do VNTS em 14 instâncias das 30 testadas. Entretanto, no tempo computacional, verificamos um aumento, o que passa a ser uma desvantagem do método que propusemos mas, acreditamos que essa desvantagem possa ser contornada se a técnica GVNS passar a ser chamada após um certo número de iterações GRASP e não a cada iteração como foi feito.

Resultados da classe $C2$

Nesta classe foram submetidas 33 instâncias a ambos os algoritmos. O GGVNS encontrou melhores soluções em 13 das 33 instâncias, como ilustrado pela coluna GAP da tabela 6.16, para $\alpha = 0,9$. O desvio padrão é baixo ao analisar o valor de GAP, reforçando a robustez da aproximação de GGVNS.

A tabela 6.17 apresenta na primeira coluna o nome das instâncias, nas segunda e terceira colunas os nomes das metaheurísticas, onde em cada uma mostramos os valores mínimos da função objetivo f_min e o tempo médio de execução t_avg, e na quarta e última coluna apresentamos o valor ótimo (opt), segundo [27].

Como podemos observar na tabela 6.17 aplicando a GGVNS ao PCMRC encontramos o valor ótimo em 21 das 33 instâncias, enquanto que na técnica desenvolvida por Pérez e outros em [34], VNTS, o valor ótimo foi alcançado em 19 instâncias. E em 5 das instâncias onde o ótimo não foi alcançado por nenhuma das duas técnicas utilizadas, a GGVNS obteve resultados melhores que a VNTS, enquanto que a GGVNS perdeu em apenas uma das instâncias onde o ótimo não foi encontrado.

	Beta	SABT		GBT		GGVNS	
instância		f_min	t_min	f_min	t_min	f_min	t_min
f10_1.rcp	5	12295	0,029	12295	0,008	12295	0,018
	7	12212	0,01	12212	0,005	12212	0,017
	9	8364	0,009	8364	0,006	8364	0,018
f10_2.rcp	5	13600	0,041	13600	0,009	13600	0,018
	7	14239	0,021	14239	0,008	14239	0,019
	9	8083	0,009	8083	0,005	8083	0,016
f10_3.rcp	5	12315	0,04	12315	0,007	12315	0,015
	7	12307	0,01	12307	0,005	12307	0,017
	9	8459	0,009	8459	0,004	8459	0,017
f10_4.rcp	5	14485	0,041	14485	0,006	14485	0,015
	7	15247	0,009	15247	0,004	15247	0,02
	9	11029	0,009	11029	0,004	11029	0,019
f10_5.rcp	5	14705	0,016	14705	0,007	14705	0,014
	7	16754	0,01	16754	0,006	16754	0,017
	9	11338	0,008	11338	0,004	11338	0,017
f10_6.rcp	5	15495	0,029	15495	0,006	15535	0,016
	7	15521	0,021	15521	0,005	15521	0,018
	9	9420	0,010	9420	0,006	9420	0,017
f10_7.rcp	5	14455	0,041	14455	0,007	14455	0,016
	7	15282	0,009	15282	0,005	15282	0,018
	9	10926	0,01	10926	0,005	11254	0,017
f10_8.rcp	5	13525	0,042	13525	0,007	13525	0,016
	7	13929	0,013	13929	0,007	13929	0,018
	9	10323	0,011	10323	0,005	10323	0,018
f10_9.rcp	5	16475	0,04	16475	0,005	16475	0,016
	7	14684	0,01	14684	0,004	15178	0,016
	9	10888	0,009	10888	0,004	11846	0,016
f10_10.rcp	5	13050	0,039	13050	0,008	13050	0,018
	7	13574	0,01	13574	0,005	13574	0,016
	9	8898	0,009	8898	0,004	8898	0,017

Tabela 6.1: Comparação entre SABT, GBT e GGVNS para instâncias com 10 vértices

	Beta	SABT		GBT		GGVNS	
instância		f_min	t_min	f_min	t_min	f_min	t_min
f20_1.rcp	5	16195	0,213	16945	0,076	16195	0,215
	7	16295	0,134	16295	0,043	16295	0,273
	9	9100	0,041	9100	0,031	9100	0,182
f20_2.rcp	5	17075	0,217	17075	0,093	17075	0,189
	7	19946	0,178	19946	0,053	19946	0,309
	9	12085	0,036	12085	0,020	12125	0,185
f20_3.rcp	5	16275	0,186	16275	0,095	16275	0,125
	7	19444	0,168	19444	0,048	19444	0,289
	9	15967	0,039	15967	0,034	15967	0,201
f20_4.rcp	5	19375	0,100	19315	0,089	19315	0,111
	7	20517	0,124	20517	0,044	20517	0,267
	9	11540	0,036	11540	0,022	11540	0,174
f20_5.rcp	5	16480	0,204	16140	0,063	16140	0,247
	7	16298	0,171	16298	0,045	16298	0,303
	9	8835	0,056	8835	0,048	8835	0,181
f20_6.rcp	5	18045	0,211	18135	0,072	18045	0,207
	7	19818	0,171	19818	0,049	19818	0,304
	9	12721	0,038	12721	0,034	12721	0,180
f20_7.rcp	5	17305	0,172	17305	0,071	17305	0,169
	7	20520	0,174	20520	0,060	20520	0,264
	9	11581	0,039	11581	0,036	11581	0,183
f20_8.rcp	5	16045	0,191	16045	0,077	16045	0,178
	7	18615	0,164	18615	0,052	18615	0,325
	9	10860	0,039	10860	0,035	10860	0,194
f20_9.rcp	5	16085	0,199	16085	0,075	16085	0,215
	7	17202	0,18	17202	0,062	17202	0,281
	9	11352	0,036	11352	0,037	11352	0,178
f20_10.rcp	5	18420	0,203	18420	0,094	18420	0,194
	7	19688	0,189	19688	0,071	19688	0,308
	9	12535	0,045	12535	0,034	12535	0,194

Tabela 6.2: Comparação entre SABT, GBT e GGVNS para instâncias com 20 vértices

	Beta	SABT		GBT		GGVNS	
instância		f_min	t_min	f_min	t_min	f_min	t_avg
f30_1.rcp	5	19605	0,55	19605	0,30	19605	0,896
	7	22578	0,51	22578	0,17	22578	1,596
	9	14052	0,08	14052	0,67	14052	0,724
f30_2.rcp	5	19635	0,59	19665	0,26	19635	1,143
	7	22458	0,54	22458	0,21	22458	1,578
	9	16292	0,10	16292	0,08	16292	0,705
f30_3.rcp	5	21335	0,57	21335	0,31	21335	0,936
	7	24985	0,47	24960	0,14	24793	1,564
	9	13435	0,08	13435	0,06	13435	0,682
f30_4.rcp	5	18040	0,56	18040	0,29	18040	1,061
	7	20580	0,46	20460	0,15	20460	1,466
	9	12574	0,72	12574	0,09	12574	0,678
f30_5.rcp	5	21520	0,55	21520	0,35	21520	0,992
	7	24825	0,34	24825	0,15	24825	1,324
	9	16701	0,08	16701	0,08	16854	0,646
f30_6.rcp	5	21905	0,55	21905	0,33	21905	0,841
	7	24807	0,51	24807	0,16	24807	1,529
	9	16708	0,09	16708	0,07	16708	0,677
f30_7.rcp	5	19655	0,59	19655	0,36	19655	1,049
	7	22648	0,51	22383	0,19	22387	1,689
	9	15368	0,21	15368	0,09	15368	0,968
f30_8.rcp	5	22980	0,52	22220	0,27	22220	1,166
	7	22440	0,43	22440	0,14	22440	1,548
	9	13401	0,82	13401	0,06	13401	0,764
f30_9.rcp	5	19845	0,63	19845	0,32	19845	0,984
	7	22718	0,55	22718	0,18	22718	1,635
	9	15026	0,07	15026	0,08	15026	0,690
f30_10.rcp	5	19675	0,91	19675	0,30	19675	1,250
	7	22791	0,76	22791	0,16	22791	1,650
	9	15279	0,10	15279	0,07	15279	0,797

Tabela 6.3: Comparação entre SABT, GBT e GGVNS para instâncias com 30 vértices

	Beta	SABT		GBT		GGVNS	
instâncias		f_min	t_min	f_min	t_min	f_min	t_min
f40_1.rcp	5	24360	1,43	24360	0,65	24360	4,374
	7	27431	0,76	27431	0,30	27431	5,464
	9	17792	0,16	17792	0,13	17792	1,887
f40_2.rcp	5	22940	1,33	22940	0,62	22940	4,617
	7	25262	0,95	25255	0,38	25255	5,624
	9	17590	0,14	17590	0,13	17590	1,778
f40_3.rcp	5	23545	1,345	23545	0,70	23545	4,697
	7	26142	0,94	25825	0,36	25789	5,158
	9	17530	0,32	17530	0,15	17530	2,949
f40_4.rcp	5	24665	1,40	24585	0,74	24585	4,393
	7	28566	1,13	27517	0,39	27517	5,949
	9	17145	0,19	17145	0,14	17145	1,865
f40_5.rcp	5	24725	1,40	24345	0,83	24345	4,245
	7	28268	1,06	28329	0,44	28276	5,540
	9	21159	0,63	21159	0,15	21159	3,810
f40_6.rcp	5	24380	1,28	24380	0,78	24380	3,454
	7	28386	1,11	27480	0,36	27491	5,880
	9	19143	0,43	19143	0,16	19143	2,014
f40_7.rcp	5	21905	1,41	21905	0,80	21905	3,998
	7	26026	1,23	25973	0,44	25973	5,961
	9	21584	0,47	21584	0,19	21584	3,362
f40_8.rcp	5	24400	1,45	23665	0,75	23665	4,580
	7	28279	1,26	26830	0,38	26842	5,272
	9	18461	0,55	18461	0,14	18461	3,315
f40_9.rcp	5	25255	1,35	24980	0,79	24980	3,956
	7	29393	1,05	29393	0,42	29393	5,067
	9	24120	0,44	24120	0,17	24120	2,883
f40_10.rcp	5	24395	1,37	24395	0,61	24395	4,451
	7	26323	1,13	26323	0,36	26323	5,030
	9	17400	0,16	17400	0,15	17400	3,048

Tabela 6.4: Comparação entre SABT, GBT e GGVNS para instâncias com 40 vértices

	Beta	SABT		GBT		GGVNS	
instância		f_min	t_min	f_min	t_min	f_min	t_min
f50_1.rcp	5	28145	3,20	28100	1,52	28100	12,311
	7	30020	2,49	30020	0,80	30020	13,209
	9	22942	0,68	22942	0,23	22942	5,723
f50_2.rcp	5	29345	2,71	28545	1,73	28510	10,525
	7	30629	1,82	30629	0,72	30629	13,846
	9	22081	0,49	22081	0,29	22081	5,217
f50_3.rcp	5	30055	2,19	28610	1,65	28535	11,086
	7	32607	1,78	32275	0,71	32129	13,290
	9	22295	0,75	22300	0,29	22300	6,648
f50_4.rcp	5	24735	2,04	24735	1,46	24735	9,903
	7	29009	1,73	28840	0,80	28788	14,327
	9	20778	0,29	20778	0,23	20778	6,842
f50_5.rcp	5	26950	2,27	26950	1,27	26950	14,167
	7	29268	1,70	29268	0,83	29268	15,425
	9	23585	0,77	23585	0,30	23585	8,588
f50_6.rcp	5	26225	2,13	26295	1,27	26225	13,163
	7	30206	1,64	29881	0,70	29841	15,287
	9	20709	0,65	20709	0,29	20709	6,636
f50_7.rcp	5	26095	2,04	26095	1,41	26095	13,646
	7	29901	2,06	30030	0,77	29901	15,295
	9	22127	0,40	22127	0,24	22127	4,245
f50_8.rcp	5	27505	2,14	27320	1,58	27195	10,500
	7	33423	1,47	32603	0,83	32603	14,235
	9	26564	0,72	26564	0,28	26564	7,380
f50_9.rcp	5	27055	2,32	26970	1,29	26900	12,388
	7	29438	1,62	29261	0,81	29261	12,196
	9	22282	0,78	22282	0,32	22282	7,370
f50_10.rcp	5	27890	2,16	27925	1,41	27890	12,627
	7	28651	1,34	28651	0,75	28651	13,428
	9	21715	0,22	21715	0,21	21715	3,964

Tabela 6.5: Comparação entre SABT, GBT e GGVNS para instâncias com 50 vértices

	Beta	SABT		GBT		GGVNS	
instância		f_min	t_min	f_min	t_min	f_min	t_min
f75_1.rcp	5	33880	8,61	33520	6,21	33520	94,67
	7	37977	6,88	38032	3,01	37920	80,99
	9	27479	1,93	27479	0,78	27479	28,13
f75_2.rcp	5	32495	8,12	31080	6,50	30745	85,08
	7	35298	6,61	35190	2,50	35032	84,14
	9	26972	2,53	26972	0,86	26972	38,60
f75_3.rcp	5	35105	8,02	33345	6,19	33100	86,14
	7	38216	7,42	37451	2,81	37098	82,1
	9	28013	2,08	27388	0,83	27388	34,85
f75_4.rcp	5	31415	7,80	31660	5,61	31285	88,51
	7	36123	5,73	34860	2,66	34583	77,60
	9	26073	2,25	26098	0,72	25946	31,51
f75_5.rcp	5	30320	7,83	29885	5,82	29840	101,92
	7	34173	6,54	33726	3,42	33734	115,41
	9	29114	3,21	28770	1,23	28783	31,17
f75_6.rcp	5	33310	7,64	33170	7,24	32420	86,74
	7	39155	6,22	37122	3,23	37122	104,54
	9	25962	1,57	25962	0,81	26034	33,81
f75_7.rcp	5	33280	8,48	32020	6,78	31900	90,12
	7	36429	5,94	35964	3,89	35381	81,17
	9	28665	2,95	28192	1,08	28178	52,43
f75_8.rcp	5	31355	8,47	31005	7,41	30900	85,51
	7	35888	11,30	35464	2,80	35356	80,06
	9	26543	3,36	26543	0,96	26575	43,71
f75_9.rcp	5	32110	11,93	31175	6,40	30825	131,31
	7	34265	10,65	34072	3,37	34028	94,51
	9	24957	2,75	24957	1,10	24957	40,65
f75_10.rcp	5	31725	12,86	31070	6,48	30930	90,70
	7	35389	5,82	35004	2,79	34728	80,30
	9	28382	3,29	28448	1,07	28382	48,06

Tabela 6.6: Comparação entre SABT, GBT e GGVNS para instâncias com 75 vértices

	Beta	SABT		GBT		GGVNS	
instância		f_min	t_min	f_min	t_min	f_min	t_min
f100_1.rcp	5	37670	23,98	38545	15,25	36665	338,8
	7	42027	14,52	42456	5,31	41279	313,4
	9	29543	6,33	29565	1,87	29543	113,9
f100_2.rcp	5	38760	20,91	38140	15,4	37500	349,6
	7	42835	18,68	42169	6,98	41983	288,1
	9	31198	7,83	31155	2,30	31166	125,8
f100_3.rcp	5	36255	27,64	35200	13,40	34750	372,8
	7	38432	18,02	38571	5,89	37912	279,6
	9	28180	5,75	28190	2,36	28180	148
f7100_4.rcp	5	39775	29,36	37895	14,34	37345	382,1
	7	43365	27,00	42233	6,23	41907	305,6
	9	33956	9,30	33452	2,05	33452	117,6
f100_5.rcp	5	36245	32,02	35925	20,46	35730	386,8
	7	40964	23,34	41395	8,57	40338	305
	9	33818	8,35	33562	2,92	33451	144,8
f100_6.rcp	5	38495	19,80	36400	17,07	36355	380,8
	7	42290	20,73	41677	8,83	41743	283,3
	9	32005	7,44	31677	1,83	31677	111,6
f100_7.rcp	5	36465	25,41	36225	13,60	35840	385,5
	7	39580	18,84	39234	6,14	38950	293
	9	29926	6,71	29946	1,60	29911	129,8
f100_8.rcp	5	38800	22,94	38000	16,59	37575	377,4
	7	42872	14,49	43250	6,69	42463	285,7
	9	31236	5,45	31016	2,10	30789	176,2
f100_9.rcp	5	38695	21,96	37025	19,01	36780	378,2
	7	41798	17,15	40636	8,61	40372	278,9
	9	31828	7,80	31828	3,06	31828	159
f100_10.rcp	5	39955	22,28	38330	18,81	38355	376,4
	7	44355	28,27	43549	8,03	43130	280
	9	31465	8,76	31465	2,12	31425	128,3

Tabela 6.7: Comparação entre SABT, GBT e GGVNS para instâncias com 100 vértices

n	SABT	GBT	GGVNS
10	0/30	0/30	4/30
20	2/30	1/30	1/30
30	4/30	2/30	2/30
40	9/30	3/30	2/30
50	8/30	9/30	2/30
75	24/30	20/30	4/30
100	27/30	24/30	3/30
Total	74/210	59/210	18/210

Tabela 6.8: Comparação entre o número de derrotas das metaheurística SABT, GBT e GGVNS em relação as outras duas.

Instância	β	GGVNS		VNTS		GAP
		f_min	t_avg	f_min	t_avg	
f10_1.rcp	5	12295	0,019	12295	0,012	0
	7	12212	0,018	12212	0,007	0
	9	8364	0,018	8364	0,008	0
f10_2.rcp	5	13600	0,019	13600	0,012	0
	7	14239	0,019	14239	0,009	0
	9	8083	0,017	8083	0,007	0
f10_3.rcp	5	12315	0,015	12315	0,011	0
	7	12307	0,017	12307	0,007	0
	9	8459	0,018	8459	0,007	0
f10_4.rcp	5	14485	0,016	14485	0,010	0
	7	15247	0,023	15247	0,008	0
	9	11029	0,019	11029	0,007	0
f10_5.rcp	5	14705	0,015	14705	0,010	0
	7	16754	0,021	16754	0,007	0
	9	11338	0,017	11338	0,008	0
f10_6.rcp	5	15535	0,019	15495	0,012	-0,002575
	7	15521	0,021	15521	0,009	0
	9	9420	0,017	9420	0,007	0
f10_7.rcp	5	14455	0,016	14455	0,013	0
	7	15282	0,018	15282	0,009	0
	9	11254	0,018	10926	0,008	-0,029145
f10_8.rcp	5	13525	0,017	13525	0,012	0
	7	13929	0,022	13929	0,009	0
	9	10323	0,018	10323	0,007	0
f10_9.rcp	5	16475	0,016	16475	0,010	0
	7	15178	0,017	14684	0,008	-0,032547
	9	11846	0,016	10888	0,007	-0,080871
f10_10.rcp	5	13050	0,018	13050	0,011	0
	7	13574	0,017	13574	0,009	0
	9	8898	0,018	8898	0,007	0

Tabela 6.9: Comparação entre GGVNS e VNTS para instâncias de classe C1 com 10 vértices.

Instância	β	GGVNS		VNTS		GAP
		f_min	t_avg	f_min	t_avg	
f20_1.rcp	5	16195	0,216	16195	0,146	0
	7	16295	0,280	16295	0,095	0
	9	9100	0,186	9100	0,043	0
f20_2.rcp	5	17075	0,190	17075	0,126	0
	7	19946	0,316	19946	0,101	0
	9	12125	0,194	12085	0,038	-0,003299
f20_3.rcp	5	16275	0,125	16275	0,147	0
	7	19444	0,291	19444	0,104	0
	9	15967	0,211	15967	0,041	0
f20_4.rcp	5	19315	0,113	19315	0,069	0
	7	20517	0,287	20517	0,094	0
	9	11540	0,195	11540	0,038	0
f20_5.rcp	5	16140	0,252	16140	0,095	0
	7	16298	0,321	16298	0,101	0
	9	8835	0,184	8835	0,041	0
f20_6.rcp	5	18045	0,213	18121	0,125	0,004212
	7	19818	0,310	19818	0,096	0
	9	12721	0,195	12721	0,039	0
f20_7.rcp	5	17305	0,178	17389	0,064	0,004854
	7	20520	0,269	20520	0,080	0
	9	11581	0,190	11581	0,038	0
f20_8.rcp	5	16045	0,183	16045	0,123	0
	7	18615	0,327	18615	0,093	0
	9	10860	0,195	10860	0,037	0
f20_9.rcp	5	16085	0,220	16085	0,100	0
	7	17202	0,284	17202	0,091	0
	9	11352	0,183	11352	0,040	0
f20_10.rcp	5	18420	0,199	18420	0,124	0
	7	19688	0,313	19688	0,098	0
	9	12535	0,201	12535	0,043	0

Tabela 6.10: Comparação entre GGVNS e VNTS para instâncias de classe C1 com 20 vértices.

Instância	β	GGVNS		VNTS		GAP
		f_min	t_avg	f_min	t_avg	
f30_1.rcp	5	19605	0,949	19605	0,279	0
	7	22578	1,658	22578	0,376	0
	9	14052	0,711	14052	0,090	0
f30_2.rcp	5	19635	1,228	19635	0,252	0
	7	22458	1,626	22458	0,433	0
	9	16292	0,710	16292	0,094	0
f30_3.rcp	5	21335	1,030	21685	0,289	0,016405
	7	24793	1,582	24793	0,376	0
	9	13435	0,703	13435	0,093	0
f30_4.rcp	5	18040	1,151	18040	0,563	0
	7	20460	1,488	20460	0,405	0
	9	12574	0,705	12574	0,104	0
f30_5.rcp	5	21520	1,019	21536	0,266	0,000743
	7	24825	1,378	24825	0,406	0
	9	16854	0,744	16701	0,104	-0,009078
f30_6.rcp	5	21905	0,889	21953	0,234	0,002191
	7	24807	1,577	24807	0,457	0
	9	16708	0,701	16708	0,104	0
f30_7.rcp	5	19655	1,089	19655	0,348	0
	7	22387	1,730	22383	0,326	-0,000179
	9	15368	1,000	15368	0,126	0
f30_8.rcp	5	22220	1,195	22259	0,258	0,001755
	7	22440	1,663	22440	0,415	0
	9	13401	0,807	13401	0,099	0
f30_9.rcp	5	19845	1,054	19845	0,530	0
	7	22718	1,699	22718	0,405	0
	9	15026	0,806	15026	0,097	0
f30_10.rcp	5	19675	1,303	19747	0,434	0,003659
	7	22791	1,690	22791	0,452	0
	9	15279	0,858	15279	0,123	0

Tabela 6.11: Comparação entre GGVNS e VNTS para instâncias de classe C1 com 30 vértices.

Instância	β	GGVNS		VNTS		GAP
		f_min	t_avg	f_min	t_avg	
f40_1.rcp	5	24360	4,480	24360	0,718	0
	7	27431	5,645	27431	1,143	0
	9	17792	1,985	17792	0,189	0
f40_2.rcp	5	22940	4,708	22940	0,573	0
	7	25255	5,717	25255	1,044	0
	9	17590	1,993	17590	0,193	0
f40_3.rcp	5	23545	4,776	23545	0,626	0
	7	25789	5,371	25789	1,086	0
	9	17530	3,025	17530	0,288	0
f40_4.rcp	5	24585	4,485	25333	0,667	0,030425
	7	27517	5,978	27517	1,025	0
	9	17145	2,005	17145	0,173	0
f40_5.rcp	5	24345	4,495	24345	0,554	0
	7	28276	5,682	28268	1,039	-0,000283
	9	21159	3,894	21159	1,021	0
f40_6.rcp	5	24380	3,914	24380	0,539	0
	7	27491	6,005	27480	1,074	-0,000400
	9	19143	2,238	19143	0,234	0
f40_7.rcp	5	21905	4,055	22185	0,583	0,012782
	7	25973	6,059	25973	1,121	0
	9	21584	3,436	21584	0,625	0
f40_8.rcp	5	23665	4,766	23690	0,717	0,001056
	7	26842	5,391	26830	0,734	-0,000447
	9	18461	3,398	18461	0,769	0
f40_9.rcp	5	24980	4,268	24980	0,517	0
	7	29393	5,423	29393	1,109	0
	9	24120	2,989	24120	0,691	0
f40_10.rcp	5	24395	4,614	24639	1,158	0,010002
	7	26323	5,218	26323	0,702	0
	9	17400	3,170	17400	0,295	0

Tabela 6.12: Comparação entre GGVNS e VNTS para instâncias de classe C1 com 40 vértices.

Instância	β	GGVNS		VNTS		GAP
		f_min	t_avg	f_min	t_avg	
f50_1.rcp	5	28100	12,943	28100	1,127	0
	7	30020	13,789	30020	1,688	0
	9	22942	5,887	22942	1,206	0
f50_2.rcp	5	28510	10,908	28545	1,074	0,001228
	7	30629	14,086	30629	1,728	0
	9	22081	5,377	22081	0,815	0
f50_3.rcp	5	28535	11,226	28610	1,189	0,002628
	7	32129	13,690	32129	1,889	0
	9	22300	6,901	22295	1,573	-0,000224
f50_4.rcp	5	24735	10,772	24811	1,071	0,003072
	7	28788	14,571	28788	2,286	0
	9	20778	7,164	20778	0,916	0
f50_5.rcp	5	26950	14,622	27135	1,165	0,006864
	7	29268	15,726	29268	2,493	0
	9	23585	8,695	23585	1,344	0
f50_6.rcp	5	26225	13,620	26425	1,154	0,007626
	7	29841	15,474	29841	2,373	0
	9	20709	6,870	20709	1,332	0
f50_7.rcp	5	26095	14,376	26095	1,136	0
	7	29901	16,180	29901	2,100	0
	9	22127	4,653	22127	0,425	0
f50_8.rcp	5	27195	10,671	27195	1,094	0
	7	32603	14,429	32603	1,882	0
	9	26564	7,496	26564	2,300	0
f50_9.rcp	5	26900	12,761	26937	1,169	0,001375
	7	29261	12,392	29261	2,174	0
	9	22282	7,532	22282	1,367	0
f50_10.rcp	5	27890	13,016	27880	1,191	-0,000358
	7	28651	13,831	28651	2,463	0
	9	21715	4,379	21715	0,340	0

Tabela 6.13: Comparação entre GGVNS e VNTS para instâncias de classe C1 com 50 vértices.

Instância	β	GGVNS		VNTS		GAP
		f_min	t_avg	f_min	t_avg	
f75_1.rcp	5	33520	104,61	33730	3,85	0,006265
	7	37920	84,87	37920	4,79	0
	9	27479	28,88	27479	5,44	0
f75_2.rcp	5	30745	98,26	31100	3,91	0,011547
	7	35032	89,07	35032	3,85	0
	9	26972	39,78	26972	4,08	0
f75_3.rcp	5	33100	89,53	33705	4,09	0,018278
	7	37098	84,00	37096	4,32	-0,000054
	9	27388	36,16	27388	3,93	0
f75_4.rcp	5	31285	92,48	31517	4,03	0,007416
	7	34583	77,79	34583	6,38	0
	9	25946	33,84	25855	3,79	-0,003507
f75_5.rcp	5	29840	104,71	29906	3,94	0,002212
	7	33734	116,07	33726	3,98	-0,000237
	9	28783	38,57	28770	6,35	-0,000452
f75_6.rcp	5	32420	87,96	32696	3,91	0,008513
	7	37122	105,29	37188	4,67	0,001778
	9	26034	40,84	25962	4,01	-0,002766
f75_7.rcp	5	31900	134,34	32067	3,93	0,005235
	7	35381	82,33	35389	4,13	0,000226
	9	28178	53,45	28178	9,45	0
f75_8.rcp	5	30900	100,67	31271	4,64	0,012006
	7	35356	81,16	35356	5,65	0
	9	26575	55,09	26543	5,97	-0,001204
f75_9.rcp	5	30825	147,92	30825	4,10	0
	7	34028	106,91	34028	6,00	0
	9	24957	41,09	24957	7,45	0
f75_10.rcp	5	30930	94,22	31250	4,59	0,010346
	7	34728	81,77	34753	4,37	0,000720
	9	28382	49,43	28286	8,0	-0,003382

Tabela 6.14: Comparação entre GGVNS e VNTS para instâncias com 75 vertices

Instance	β	GGVNS		VNTS		GAP
		f_min	t_avg	f_min	t_avg	
f100_1.rcp	5	36665	340,9	37250	10,2	0,015955
	7	41279	363,9	41279	11,3	0
	9	29543	123,7	29543	12,5	0
f100_2.rcp	5	37500	359,7	37664	9,5	0,004373
	7	41983	321,6	41971	12,8	-0,00029
	9	31166	129,2	31115	7,8	-0,00164
f100_3.rcp	5	34750	389,2	34890	9,7	0,004029
	7	37912	325,3	38133	10,7	0,005829
	9	28180	150,9	28180	9,5	0
f7100_4.rcp	5	37345	430,8	37762	11	0,011166
	7	41907	364,8	41752	10,1	-0,0037
	9	33452	122,2	33452	14	0
f100_5.rcp	5	35730	447,5	35958	10,6	0,006381
	7	40338	354,7	40802	11,3	0,011503
	9	33451	146,4	33451	25,1	0
f100_6.rcp	5	36355	418,2	36823	9,4	0,012873
	7	41743	319,2	41772	10,3	0,000695
	9	31677	158,9	31612	14,5	-0,00205
f100_7.rcp	5	35840	392,9	35984	9,8	0,004018
	7	38950	310,3	38910	9,9	-0,00103
	9	29911	134	29911	7,5	0
f100_8.rcp	5	37575	396,2	37885	10,5	0,00825
	7	42463	319,4	42553	11	0,002119
	9	30789	188,3	30779	25,6	-0,00032
f100_9.rcp	5	36780	427,6	37242	10,5	0,012561
	7	40372	324,1	40372	11,3	0
	9	31828	168,1	31801	9,3	-0,00085
f100_10.rcp	5	38355	409,5	38509	9,8	0,004015
	7	43130	316,6	43074	10,4	-0,0013
	9	31425	147	31425	13,6	0

Tabela 6.15: Comparação entre GGVNS e VNTS para instâncias de classe C1 com 100 vértices.

Instância	GGVNS				VNTS				GAP	β
	f_min	f_avg	f_max	t_avg	f_min	f_avg	f_max	t_avg		
eil51.rcp	1995	1998	2005	12,35	2025	2040,2	2059	1,27	0,015038	5
	2113	2117,4	2123	10,11	2113	2113	2113	1,76	0	7
	1244	1244	1244	4,98	1244	1244	1244	0,98	0	9
berlin52.rcp	36115	36152	36260	16,80	36720	37326,8	38259	1,40	0,016752	5
	37376	37447,2	37548	16,26	37376	37675	38446	2,05	0	7
	20361	20361	20361	7,41	20361	20361	20361	1,02	0	9
eil76.rcp	2460	2461	2465	91,10	2507	2511,2	2515	4,11	0,019106	5
	2504	2504	2504	83,70	2504	2504	2504	5,68	0	7
	1710	1710	1710	30,14	1710	1710	1710	6,17	0	9
pr76.rcp	500395	500403	500405	133,81	502225	509954	513713	4,21	0,003657	5
	556939	557755,8	558555	74,643	555858	557953,4	559760	4,67	-0,001941	7
	424359	424359	424359	31,22	424359	424661	425114	3,94	0	9
rat99.rcp	5885	5898	5915	331,90	5997	6094,2	6160	9,92	0,019031	5
	6447	6460,8	6476	229,20	6436	6442	6451	8,92	-0,001706	7
	5150	5158,8	5164	140,50	5150	5151,6	5158	8,25	0	9
rd100.rcp	37975	38006	38060	330,45	38447	38872,6	39175	10,58	0,012429	5
	40952	41033	41131	237,33	40971	41046,2	41347	10,18	0,000464	7
	31776	31780,8	31784	120,95	31776	31776	31776	9,49	0	9
kroA100.rcp	100785	100797	100845	321,652	101230	103126,6	105305	10,25	0,004415	5
	115438	115519,4	115607	282,35	115388	117095,4	120601	11,82	-0,000433	7
	94467	94574,4	94852	101,00	94265	94652,6	95697	8,03	-0,002138	9
kroB100.rcp	104575	104758	104880	350,46	105073	107006	108159	10,75	0,004762	5
	118112	118481,4	118661	240,84	118183	119392,8	122302	9,43	0,000601	7
	94018	94047,6	94055	87,48	93938	93965,6	94026	8,04	-0,000851	9
kroC100.rcp	99570	99588	99600	318,58	99940	100747,6	102000	10,25	0,003716	5
	113533	113566	113698	266,27	113533	113533	113533	11,57	0	7
	92894	92894	92894	169,62	92894	92894	92894	25,92	0	9
kroD100.rcp	101795	101896	102115	337,98	103998	104948,8	106275	9,97	0,021543	5
	117297	117525,8	117814	304,49	116924	118105,2	121016	9,84	-0,003180	7
	92225	92249,8	92349	129,51	92102	92102	92102	18,95	-0,001334	9
kroE100.rcp	104915	105079	105220	341,77	105003	105693,4	106547	10,67	0,000839	5
	116471	116471	116471	253,31	116471	117562	119027	10,91	0	7
	96116	96281,2	96346	116,10	96116	96119,6	96122	107,07	0	9

Tabela 6.16: Comparação entre GGVNS e VNTS para instâncias de classe C2.

Instância	β	GGVNS		VNTS		opt
		f_min	t_avg	f_min	t_avg	
eil51.rcp	5	1995	12,35	2025	1,27	1995
	7	2113	10,11	2113	1,76	2113
	9	1244	4,98	1244	0,98	1244
berlin52.rcp	5	36115	16,80	36720	1,40	36115
	7	37376	16,26	37376	2,05	37376
	9	20361	7,41	20361	1,02	20361
eil76.rcp	5	2460	91,10	2507	4,11	2460
	7	2504	83,70	2504	5,68	2504
	9	1710	30,14	1710	6,17	1710
pr76.rcp	5	500395	133,81	502225	4,21	500395
	7	556939	74,643	555858	4,67	555858
	9	424359	31,22	424359	3,94	424359
rat99.rcp	5	5885	331,90	5997	9,92	5885
	7	6447	229,20	6436	8,92	6436
	9	5150	140,50	5150	8,25	5150
rd100.rcp	5	37975	330,45	38447	10,58	37975
	7	40952	237,33	40971	10,18	40915
	9	31776	120,95	31776	9,49	31776
kroA100.rcp	5	100785	321,652	101230	10,25	100785
	7	115438	282,35	115388	11,82	115388
	9	94467	101,00	94265	8,03	94265
kroB100.rcp	5	104575	350,46	105073	10,75	104550
	7	118112	240,84	118183	9,43	118111
	9	94018	87,48	93938	8,04	93938
kroC100.rcp	5	99570	318,58	99940	10,25	99065
	7	113533	266,27	113533	11,57	113533
	9	92894	169,62	92894	25,92	92894
kroD100.rcp	5	101795	337,98	103998	9,97	101645
	7	117297	304,49	116924	9,84	116849
	9	92225	129,51	92102	18,95	92102
kroE100.rcp	5	104915	341,77	105003	10,67	104915
	7	116471	253,31	116471	10,91	116471
	9	96116	116,10	96116	107,07	96116

Tabela 6.17: Comparação entre GGVNS, VNTS e os valores ótimos para instâncias de classe C2.

Capítulo 7

Conclusões e sugestões para trabalhos futuros

Neste trabalho apresentamos propostas de heurísticas para serem incorporadas aos métodos *Simulated Annealing*, Busca Tabu e GRASP para o Problema do Ciclo Mediano sem Res-trições de Capacidade (PCMRC). Após testes computacionais preliminares verificamos que algumas dessas heurísticas puras não obtiveram nenhum melhoria quando comparadas aos resultados da metaheurística VNTS, como foi o e-xemplo da Busca Tabu, já o GRASP conseguiu algumas melhorias, mas numa quantidade muito baixa. E devido ao fraco desempenho apresentado por essas heurísticas básicas, resolvemos testar metaheurísticas híbridas. Apresentamos assim as metaheurísticas híbridas SABT, GBT e GGVNS para o PCMRC, onde usamos o algoritmo busca tabu com lista dinâmica e estratégia de oscilação, nas duas primeiras metaheurísticas. Cada um dos algoritmos propostos foi executado 5 vezes sobre cada uma das 243 instâncias testadas.

Na fase construtiva dos algoritmos propostos para o PCMRC utilizamos uma heurística construtiva da melhor inserção no ciclo. Já na fase de busca

local utilizamos cinco vizinhanças, adição, remoção e troca de vértices, além das trocas de duas e três arestas no ciclo.

Analisando os resultados obtidos com a aplicação das técnicas propostas pudemos notar que o GRASP sempre que associado a uma heurística cuja fase de busca local expanda a vizinhança, têm obtido resultados muito bons, foi o que aconteceu quando associamos o GRASP ao GVNS e a Busca Tabu com lista dinâmica e estratégia de oscilação, superando a combinação *Simulated Annealing* + Busca Tabu. Porém dentre as técnicas associadas ao GRASP a que apresentou resultados mais satisfatórios foi a metaheurística GGVNS que quando aplicada para instâncias com mais de 30 vértices conseguiu resultados de excelente qualidade, apesar do aumento crescente do tempo.

A partir dessa conclusão, comparamos os resultados obtidos com a técnica sugerida por Pérez e outros em [34] e pudemos verificar que realmente a nossa técnica tem conseguido resultados muito satisfatórios, pois usando a GGVNS conseguimos melhorar os valores de 43 das 210 instâncias testadas em relação a metaheurística VNTS para as instâncias de classe C1. Não sabemos se encontramos os valores ótimos, nem quão próximo a eles estamos, mas comparativamente obtivemos uma melhoria na qualidade das soluções, mesmo que não possamos quantificá-la. Uma desvantagem encontrada foi no tempo computacional pois, como a cada iteração GRASP chamamos o GVNS, para as instâncias maiores, com mais de 30 vértices, tivemos um aumento no tempo computacional, e uma maneira de contornar essa desvantagem seria executar o GVNS após um certo número de iterações GRASP.

Também comprovamos essa eficiência do nosso método quando o executamos para as instâncias do problema do Caixeiro Viajante, onde encontramos o valor ótimo em 21 das 33 instâncias, enquanto que na técnica desenvolvida por Pérez e outros em [34], VNTS, o valor ótimo foi alcançado

em 19 instâncias. E em 5 das instâncias onde o ótimo não foi alcançado por nenhuma das duas técnicas utilizadas, a GGVNS obteve resultados melhores que a VNTS, enquanto que a GGVNS perdeu em apenas uma das instâncias onde o ótimo não foi encontrado.

Acreditamos ser possível melhorar os resultados obtidos, assim vizualizamos como possibilidade de trabalhos futuros a paralelização dessas meta-heurísticas propostas para o PCMRC numa tentativa também de reduzir os tempos computacionais para as instâncias maiores.

Bibliografia

[1] BALAS, E., "The Prize Collecting Traveling Salesman Problem", *Networks*, v. 19, pp. 621–636, 1989.

[2] BALDACCI, R., DELL'AMICO, M., GONZÁLEZ, J. S., "The Capacitated m-Ring Star Problem", Tech. Rep. No. DISMI 42, Universitá degli Studi de Modena e Reggio Emilia, 2003.

[3] BAUER, P., "The Circuit Polytopes: Facets", *Mathematics of Operations Research*, v. 22, pp. 110–145, 1997.

[4] BOAVENTURA, P. O., *Grafos: Teoria, Modelos e Aplicações.* New York, Edgar Blücher Ltda., 1996.

[5] BOLLOBÁS, B., "Modern Graph Theory", In: *Graduate Texts in Mathematics*, v. 184.

[6] BONDY, J. A., MURTY, U. S. R., *Graph Theory with Applications.* 1 ed. New York, North Holland, 1976.

[7] DIAS, T., S.GILBERTO, MACAMBIRA, E., *et al.*, "An efficient heuristic for the ring star problem", In: *5th International Workshop Experimental Algorithms.*

[8] E M. LAGUNA, F. G., *Tabu Search.* Kluwer Academic Publishers, 1997.

[9] FEO, T. A., RESENDE, M. G. C., "Greedy Randomized Adaptive Search Procedures", *Journal of Global Optimization*, v. 6, pp. 109–133, 1995.

[10] FISCHETTI, M., SALAZAR, J., TOTH, P., "The simmetric generalized travelling salesman polytope", *Networks*, v. 26, pp. 113–123, 1995.

[11] FISCHETTI, M., SALAZAR, J., TOTH, P., "A Branch-and-Cut Algorithm for the Symmetric Generalized Travelling Salesman Problem", *Operations Research*, v. 45, pp. 378–394, 1997.

[12] FISCHETTI, M., SALAZAR, J., TOTH, P., "Solving the Orienteering Problem Through Branch-and-Cut", *INFORMS Journal on Computing*, v. 10, pp. 133–148, 1998.

[13] FOULDS, L., WILSON, J., YAMAGUCHI, T., "Modelling and solving central cycle problems with integer programming", *Computers ans Operations Research*, v. 31, pp. 1083–1095, 2004.

[14] GENDREAU, M., "An Introduction to Tabu Search", In: *Handbook of Metaheuristics* (GLOVER, F., KOCHENBERGER, G., eds.), pp. 37–54, Kluwer Academic Publishers, 2003.

[15] GENDREAU, M., LAPORTE, G., SEMET, F., "The Selective Travelling Salesman Problem", *Networks*, v. 32, pp. 263–273, 1998.

[16] GLOVER, F., "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers ans Operations Research*, v. 6, pp. 549–553, 1986.

[17] GLOVER, F., LAGUNA, M., "Tabu Search", In: *Modern Heuristic Techniques for Combinatorial Problems* (REEVES, C., ed.), pp. 70–150, Blackwell, 1993.

[18] GOURDIN, E., LABBÉ, M., YAMAN, H., "Telecommunication and Location", In: *Facility Location: applications and theory* (DREZNER, Z., HAMACHER, H., eds.), pp. 275–305, Springer, 2002.

[19] HANSEN, P., "The Steepest Ascent Mildest Descent Heuristic for Combinatoria Programming", *Science*, v. 220, pp. 671–680, 1983.

[20] HANSEN, P., MLADENOVIC, N., "Variable Neighborhood Search", *Computers and Operations Research*, v. 24, n. 11, pp. 1097–1100, 1997.

[21] HANSEN, P., MLADENOVIC, N., "Variable neighborhood search: principles and applications", *European Journal of Operational Research*, v. 130, pp. 449–467, 2001.

[22] HANSEN, P., MLADENOVIC, N., "Variable neighbourhood search", In: *Handbook of Metaheuristics* (GLOVER, F., KOCHENBERGER, G., eds.), pp. 145–184, Kluwer Academic Publishers, 2003.

[23] JOHNSON, D., MCGEOCH, L., "Experimental Analysis of Heuristics for the STSP", In: *The Traveling Salesman Problem and its Variations* (E A.P. PUNNEN, G. G., ed.), Kluwer Academic Publishers, 2002.

[24] KIRKPATRICK, S., GELLAT, D., VECCHI, M., "Optimization by Simulated Annealing", *Science*, v. 220, pp. 671–680, 1983.

[25] LABBÉ, M., LAPORTE, G., "Maximizing user Convenience and Postal Service Efficiency in Post Box Location", *Belgian Journal of Operations Research, Statistics and Computer Science*, v. 26, pp. 21–35, 1986.

[26] LABBÉ, M., LAPORTE, G., MARTÍN, I. R., "Path, tree and cycle location", pp. 187–204, Kluwer, 1998.

[27] LABBÉ, M., LAPORTE, G., MARTÍN, I. R., *et al.*, "Median Cycle Problem", Tech. Rep. No. 2001/12, Université Libre de Bruxelles, Service de Mathématiques de la Gestion, 2001.

[28] LABBÉ, M., LAPORTE, G., MARTÍN, I. R., *et al.*, "The Ring Star Problem: Polyhedral Analysis and Exact Algorithm", *Networks*, v. 43, pp. 177–189, 2004.

[29] LABBÉ, M., LAPORTE, G., MARTÍN, I. R., *et al.*, "Locating Median Cycles in Networks", *European Journal of Operational Research*, v. 160, n. 2, pp. 457–470, 2005.

[30] LEE, Y., CHIU, S., SANCHEZ, J., "A Branch and Cut Algorithm for the Steiner Ring Star Problem", *International Journal of Management Science*, v. 4, pp. 21–34, 1998.

[31] LIN, S., KERNIGHAN, B. W., "An Effective Heuristic Algorithm for the Travelling Salesman Problem", *Operations Research*, v. 21, n. 9, pp. 498–516, 1973.

[32] MARTÍN, I., *Cycle Locatin Problems*, Ph.D. dissertation, Universidad de La Laguna, La Laguna, 2000.

[33] OCHI, L., "Conhecimento heurístico: Aplicações em problemas de otimização", In: *XIV Congresso da Sociedade Brasileira de Computação.*

[34] PÉREZ, J. M., VEGA, M. M., MARTÍN, I. R., "Variable neighbourhood tabu search and its application to the median cycle problem", *European Journal of Operational Research*, v. 151, pp. 365–378, 2003.

[35] RENAUD, J., BOCTOR, F., LAPORTE, G., "Efficients Heuristics for Median Cycle Problem", *Journal of Operational Research Society*, v. 55, n. 2, pp. 179–186, 2005.

[36] RESENDE, M. G. C., RIBEIRO, C. C., "Greedy randomized adaptive search procedures", In: *Handbook of Metaheuristics* (GLOVER, F., KOCHENBERGER, G., eds.), pp. 219–249, Kluwer Academic Publishers, 2003.

[37] RESENDE, M., "Greedy Randomized Adaptive Search Procedures (GRASP)", tech. rep., AT & T Labs Research.

[38] RESENDE, M., PARDALOS, P., *Handbook of Optimization in Telecommunications*. New York, Springer Science + Business Media, 2006.

[39] SOUZA, M., *Programação de Horários em Escolas: Uma aproximação por Metaheurísticas*, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, 2000.

[40] XU, J., CHIU, S., F.GLOVER, "Optimizing a ring-based private line telecommunication network using Tabu Search", *Management Science*, v. 3, pp. 330–345, 1999.