ALGORITMOS DE OTIMIZAÇÃO PARA ROTEAMENTO E AGRUPAMENTO EM REDES DE SENSORES SEM FIO COM SORVEDOUROS MÓVEIS

CRISTIANO ARBEX VALLE

Orientador: Alexandre Salles da Cunha

ALGORITMOS DE OTIMIZAÇÃO PARA ROTEAMENTO E AGRUPAMENTO EM REDES DE SENSORES SEM FIO COM SORVEDOUROS MÓVEIS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

Julho de 2009

Valle, Cristiano Arbex
D1234p Algoritmos de Otimização para Roteamento e Agrupamento em Redes de Sensores Sem Fio com Sorvedouros Móveis / Cristiano Arbex Valle. — Belo Horizonte, 2009
xviii, 76 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Alexandre Salles da Cunha

1. Redes de Sensores Sem Fio. 2. Otimização Combinatória. 3. Pesquisa Operacional. I. Título.

CDU 519.6*82.10

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Dedico este trabalho ao meu avô Chico, ao meu tio Zé, ao meu primo Marcelo e ao Ralph.

Agradecimentos

Gostaria de agradecer aos meus pais, Joninha e Beth, e ao meu irmão, Biel (e Patrícia), pessoas que admiro muito e que me apoiaram durante toda minha vida. São responsáveis pelo meu caráter e personalidade. Não menos importantes, merecem agradecimentos, meus tios(as), primos(as) e em especial meus avós, de quem sinto grande orgulho.

Queria agradecer também à Marina, a garota que esteve ao meu lado em todos os momentos nos últimos anos, há muito parte da minha vida e da minha história. Companheira que aproveita a parte boa, que aguenta a parte ruim, que faz toda a diferença.

Aos meus amigos, parceiros em grandes histórias e casos ao longo dos anos: os brous, os amigos e amigas que conheci pela vida, o pessoal da faculdade. Mais recentemente, os amigos que fiz no LAPO e Synergia.

Agradeço também às meninas da Secretaria do DCC por darem uma aula de eficiência. Me ajudaram em tudo que foi possível e a elas sou totalmente grato.

Ao Prof. Reinaldo Vianna, meu primeiro orientador, quem me ensinou os primeiros passos no mundo da pesquisa e com quem tive imensa satisfação em trabalhar.

Aos Profs. Cid e Loureiro pelas contribuições valiosas durante minha defesa. Em especial, agradeço ao Prof. Sebastián, não apenas por também contribuir em minha defesa, mas também por ter sido essencial na minha formação durante o mestrado.

Ao Prof. Robson, meu co-orientador, sou eternamente grato por ter aberto para mim as portas do DCC, por ter depositado confiança em mim mesmo tendo sido meu professor apenas no distante ano de 2001. Se isso não bastasse, ainda contribuiu com minha formação de todas as formas possíveis, com sua vasta experiência.

Finalmente, quero agradecer ao Prof. Alexandre. Mais que orientador e professor, foi um verdadeiro parceiro durante o mestrado (e continua sendo). Com uma dedicação fora do comum, foi diretamente responsável por tudo que alcancei neste trabalho. O que ele fez por mim vai ser lembrado pra sempre.

Não são todas as pessoas que tem a sorte que eu tive de poder ter contado com tanta gente de valor durante minha vida. Ninguém alcança nada sozinho. Obrigado.

Resumo

Nesta dissertação, introduzimos modelos e algoritmos de otimização propostos para melhorar parâmetros de Qualidade de Serviço em Redes de Sensores Sem Fio com múltiplos sorvedouros móveis. Um simulador de eventos discretos, que integra os métodos de otimização propostos em um modelo realista da dinâmica da rede, também é implementado e testado computacionalmente. O principal Problema de Otimização aqui tratado, aquele de definir rotas para cada sorvedouro móvel, permitindo que os mesmos coletem informações sensoriadas da rede, é modelado como uma variante do Problema de Roteamento de Veículos não capacitado. Nesta variante, o tamanho da frota é conhecido *a priori*, nem todos os clientes devem ser visitados e o objetivo é minimizar o comprimento da maior rota. Para modelar o problema, dois Programas Inteiros são apresentados. O primeiro emprega uma formulação compacta baseada em Fluxos em Redes. Um algoritmo exato Branch-and-Bound é apresentado para esta formulação. O segundo Programa Inteiro é baseado em Desigualdades de Eliminação de Subrotas Generalizadas. Para tratar este modelo, desenvolvemos um algoritmo do tipo Branch-and-Cut. Um terceiro algoritmo, do tipo Local Branching, que emprega o método Branch-and-Cut como resolvedor interno, foi também proposto e implementado. Devido às dificuldades encontradas para resolver o problema na otimalidade com tais algoritmos, propomos também várias heurísticas baseadas em Metaheurísticas para encontrar soluções viáveis (idealmente de boa qualidade) em tempos razoáveis para os processos de decisão em Redes de Sensores Sem Fio. Nossos resultados de simulação indicam que os algoritmos de otimização permitiram alcançar melhoras significativas nas taxas de atraso na entrega de mensagens, além de obter avanços em outros parâmetros importantes de Qualidade de Serviço.

Palavras-Chave: Redes de Sensores sem Fio, Otimização Combinatória, Problema de Roteamento de Veículos, Simulação

Abstract

In this work, we introduce models and optimization algorithms to improve the Quality of Service in Wireless Sensor Networks with multiple mobile sinks. A discrete event simulator that integrates the proposed optimization methods into a realistic model of the network dynamics over the time is also implemented and tested computationaly. The main Optimization Problem considered here, that of defining routes to each mobile sink, allowing them to collect sensed information thoughout the network, is modeled as a variant of the uncapacitated Vehicle Routing Problem. In this variant, the fleet size is known beforehand, not all clients need to be visited and the goal is to minimize the length of the longest vehicle route. To model the problem, two Integer Programs are introduced. The first one is a compact formulation based on Network Flow models. A Branch-and-Bound algorithm is presented for this formulation. The second Integer Program is based in Generalized Subtour Elimination Constraints. To tackle this model, we developed a Branch-and-Cut algorithm. A third algorithm, a Local Branching that used the Branch-and-Cut as inner solver, was also proposed and implemented. Due to the difficulty found in terms of computational time in solving the problem to optimality, we also proposed several Metaheuristic based heuristics to find (hopefully) good solutions in practical times. Our simulation results indicate that the optimization algorithms allowed significant improvements in message delay rates and other important Quality of Service parameters.

Keywords: Wireless Sensor Networks, Combinatorial Optimization, Vehicle Routing Problem, Simulation

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Uma Rede de Sensores sem Fio (RSSF) é formada basicamente por um conjunto de centenas (ou mesmo milhares) de nós sensores e um ou mais sorvedouros. Um sorvedouro é um nó especial responsável por organizar a rede e coletar a informação sensoriada. Devido às suas pequenas dimensões, os nós sensores são dispositivos extremamente restritos quanto à capacidade de processamento, memória e energia disponível em sua bateria. O gerenciamento eficiente do consumo de energia talvez seja o maior desafio em aplicações envolvendo RSSFs. De acordo com Akyildiz et al. [2002], a maioria dos trabalhos encontrados na literatura que discutem a organização de uma RSSF possuem como foco principal o desenvolvimento de mecanismos que contribuam para o aumento de sua vida útil através de um controle mais eficiente de seu consumo energético.

Dadas as limitações de processamento e comunicação dos nós sensores, as informações sensoriadas necessitam ser roteadas aos sorvedouros, para que ações de controle apropriadas sejam tomadas. Em uma RSSF composta por sorvedouros cuja posição é fixa na rede, os nós sensores não poderão enviar as informações coletadas diretamente aos sorvedouros, face ao seu reduzido raio de comunicação. Ao invés disto, outros nós sensores deverão ser empregados para rotear as informações ao seu destino. Diversos autores (veja [Kim et al., 2003], [Akyildiz et al., 2002], dentre outros) argumentam que o gasto de energia quando nós sensores transmitem ou recebem informações de outros nós é superior ao gasto ao desempenhar suas funções de sensoriamento. Assim, uma estratégia que tem sido adotada com frequência para reduzir o consumo de energia da rede consiste em dotar os sorvedouros de mobilidade. Ao proceder desta forma, permite-se que o sorvedouro passeie pela RSSF, coletando informações diretamente dos sensores, sem necessariamente empregar outros nós sensores para rotear as mensagens.

A substituição da comunicação direta entre nós sensores pela comunicação sensor / sorvedouro móvel acarreta no surgimento de um novo problema. Como a velocidade de movimentação do sorvedouro é muito pequena quando comparada à velocidade de

transmissão sem fio, verifica-se um indesejado aumento no tempo decorrido entre o momento em que a informação foi sensoriada e o momento em que ela chega ao gestor da rede. Desta forma, a estratégia de dotar o sorvedouro de mobilidade para reduzir o consumo de energia tem como contraponto o atraso na entrega das mensagens.

Uma questão que surge como fruto deste conflito entre gasto de energia e atraso em RSSFs é como definir rotas *adequadas* para um ou mais sorvedouros na RSSF. Este é o tema central desta dissertação: definição de rotas para múltiplos sorvedouros em RSSFs de forma a minimizar o atraso na entrega de mensagens, sem comprometer a vida útil da rede.

Neste trabalho, este problema foi modelado como uma variação do Problema de Roteamento de Veículos (PRV) [Dantzig e Hamser, 1959]. Na variação aqui tratada, nem todo os nós sensores precisam ser visitados. Além disto, diferentemente da maioria dos estudos envolvendo o PRV (nos quais o objetivo consiste em minimizar o comprimento total das rotas), na variante aqui estudada deseja-se minimizar o comprimento da rota mais longa. Pelo que observamos com nossa revisão bibliográfica, não há estudos envolvendo esta variante do PRV.

O problema foi formalizado através de um Problema de Otimização em Grafos. A partir deste modelo, duas formulações de Programação Inteira foram propostas. A primeira delas é uma formulação compacta baseada em Fluxos em Redes [Ahuja et al., 1993]. Já a segunda emprega exponencialmente muitas Desigualdades de Eliminação de Subrotas Generalizadas (*GSEC*[1]).

Propusemos três algoritmos exatos para resolver a variante do PRV aqui estudada. O primeiro deles é um algoritmo Branch-and-Bound [Land e Doig, 1960] baseado na formulação de Fluxos. O segundo deles é baseado na formulação que emprega desigualdades *GSEC* e se trata de um algoritmo Branch-and-Cut [Grötschel et al., 1984; Padberg e Rinaldi, 1991] onde as *GSECs* são identificadas como Planos de Corte. Finalmente, o terceiro é um algoritmo do tipo Local Branching [Fischetti e Lodi, 2003] onde, como resolvedor interno, foi empregado o algoritmo Branch-and-Cut.

Uma vez que os métodos exatos aqui desenvolvidos exigem tempos elevados demais para serem utilizados em aplicações de interesse prático, desenvolvemos também métodos heurísticos de solução. Estes métodos são baseados nas metaheurísticas GRASP [Feo e Resende, 1995] e ILS [Martin e Otto, 1996].

Como parte final do trabalho, implementamos um simulador para avaliar o impacto do uso dos modelos e algoritmos propostos nas métricas mais importantes de RSSFs. O simulador não apenas implementa os algoritmos de roteamento, como também provê um método para controle de densidade, visando manter apenas um subconjunto de nós

[1]Da sigla em inglês *Generalized Subtour Elimination Constraints*

ativos que garanta total cobertura da área durante determinado período de tempo.

Os resultados obtidos via simulação com a utilização das técnicas desenvolvidas demonstraram uma considerável melhora na Qualidade de Serviço (QoS[2]) de uma RSSF. Dentre as métricas avaliadas, obtivemos importantes reduções no atraso médio na entrega de mensagens, além de uma melhora no tempo médio de vida útil da rede, quando comparados a trabalhos anteriores.

Na nossa visão, as principais contribuições deste trabalho são:

1. A forma como resolvemos o problema de roteamento em RSSFs, que resulta em um Problema de Otimização ainda pouco explorado.

2. A integração de algoritmos de otimização para resolver os problemas de roteamento e controle de densidade em um ambiente de simulação dinâmica, onde a rede e as propriedades dos nós sensores sofrem alterações com o tempo.

1.1 Organização da Dissertação

O restante desta dissertação está organizado da seguinte forma. No Capítulo 2, introduzimos alguns conceitos e aplicações de RSSFs. Uma revisão bibliográfica é apresentada, onde vários problemas relacionados à organização de uma RSSF são discutidos. Ao fim do capítulo, destacamos o problema que é tema central desta dissertação: a definição da rota de múltiplos sorvedouros móveis por uma RSSF. No Capítulo 3, formalizamos este problema, o qual denominamos Problema Integrado de Roteamento e Clusterização (PIRC), e discutimos problemas semelhantes presentes na literatura. São apresentadas ainda duas formulações de Programação Inteira para modelar o PIRC.

No Capítulo 4, introduzimos três algoritmos exatos para o PIRC. Resultados de experimentos computacionais são apresentados e as dificuldades encontradas, inerentes ao problema, são discutidas. Estas dificuldades estimularam o desenvolvimento das heurísticas propostas no Capítulo 5, que são basicamente procedimentos baseados em metaheurísticas, compostos por mecanismos de construção, diversificação e intensificação. As heurísticas implementadas são incorporadas a um simulador de eventos discretos, introduzido no Capítulo 6, para o qual são apresentados resultados de simulações realizadas. Com base nestes resultados, é possível afirmar que há melhora nos indicadores de QoS de uma RSSF devido à incorporação do PIRC e dos algoritmos que o resolvem à organização de redes deste tipo.

Finalmente, encerramos o texto no Capítulo 7, apresentando as principais conclusões extraídas deste trabalho e as direções futuras que pretendemos seguir.

[2]Sigla em inglês para *Quality of Service*

Capítulo 2

Redes de Sensores Sem Fio

Neste capítulo, apresentamos os principais conceitos envolvidos na organização de RSSFs. São apresentadas tanto aplicações práticas que justificam o interesse na área quanto especificidades e restrições que diferenciam as RSSFs de outras redes *Ad-hoc*. Em função destas especificidades, vários problemas relacionados à organização ótima de RSSFs são apresentados e discutidos. Concluímos o capítulo destacando a motivação principal desta dissertação.

2.1 Conceitos e Aplicações

Redes de Sensores Sem Fio (RSSF) são um tipo de redes *Ad-hoc*[1] baseadas no esforço colaborativo de entidades multi-funcionais autônomas e diminutas chamadas nós sensores, dotadas de:

- Função de sensoriamento para coleta de dados de eventos monitorados do meio ambiente, como temperatura e pressão, por exemplo;
- Um Processador com capacidade limitada;
- Um Rádio para comunicação sem fio;
- Uma quantidade limitada de Memória;
- Uma Bateria de baixa capacidade, que provê energia para funcionamento dos demais dispositivos.

[1]Em latim, *ad-hoc* quer dizer literalmente *apenas para este propósito*. Porém, no contexto de redes sem fio, o termo possui outro significado, denotando redes que não requerem uma infraestrutura tal como *backbones* ou pontos de acesso configurados antecipadamente.

Figura 2.1. Modelo de Nó Sensor Mica2 da linha Motes [XBOW, 2006], desenvolvida por cientistas da Universidade de Berkeley

Um nó especial, denominado sorvedouro, é responsável por uma série de atividades essenciais em RSSFs. Ele é responsável por receber e/ou processar informação e gerenciar o comportamento da rede. De modo geral, assume-se que o sorvedouro possui energia infinita (na prática, recarregável), podendo ser fixo ou móvel.

Uma RSSF é tipicamente composta por um alto número de sensores [Romer e Mattern, 2004] e um ou mais sorvedouros [Somasundara et al., 2007]. Geralmente, os nós sensores são densamente distribuídos nas proximidades dos fenômenos a serem observados, sendo que sua posição não necessita ser pré-determinada. Ao invés disto, pode ocorrer uma distribuição aleatória de sensores em terrenos inacessíveis ou em operações de resgate em áreas que sofreram algum tipo de catástrofe [Akyildiz et al., 2002].

Os nós sensores que compõem uma RSSF podem possuir distintas funções de sensoriamento. Dentre elas, podemos destacar funções sísmicas, termais, visuais, infravermelho ou acústicas. Sendo assim, uma grande variedade de eventos, fenômenos e propriedades pode ser monitorada; tais como temperatura, umidade, movimento veicular, pressão, preparação do solo, nível de ruído e a presença (ausência) de certos tipos de objetos.

Estas funcionalidades tão diversificadas, aliadas aos recentes avanços nas tecnologias de comunicação sem fio, tornam possíveis uma vasta gama de aplicações para tais redes. Segundo Akyildiz et al. [2002], algumas aplicações de destaque são:

- Operações militares: RSSFs podem ser utilizadas para ajudar na fiscalização de campos de batalha, no reconhecimento de terreno e de forças inimigas, detecção

de ataques biológicos, entre outras aplicações.

- Meio ambiente: Detecção e prevenção de acidentes sísmicos, químicos, incêndios e enchentes, assim como Mapeamento de biodiversidade em uma determinada área. As RSSFs também podem ajudar a identificar o tipo, a concentração e a localização de poluentes.

- Saúde: Os dados fisiológicos de um paciente podem ser monitorados à distância por uma equipe médica, permitindo uma melhor compreensão da condição de saúde do paciente.

- Automação de residências, controle do ambiente em prédios comerciais e monitoramento de veículos, entre outros.

Observe que as potenciais aplicações são bastante distintas entre si e possuem, cada uma, suas próprias especificidades. O projeto de uma RSSF (i.e., a definição ótima de sua topologia, protocolos de comunicação, etc.) é fortemente dependente da aplicação a que se destina, podendo ser influenciado por diversos fatores e pré-requisitos operacionais. Dentre estes fatores, podemos citar: a necessidade de tolerância a falhas, a escalabilidade de seu desempenho com o aumento da dimensão da rede (número de nós sensores e sorvedouros), os custos envolvidos, o ambiente operacional, a topologia da rede, restrições de equipamento e meios de transmissão e consumo de energia. A seguir discutimos alguns destes aspectos importantes a serem observados no projeto de RSSFs.

2.2 Organização de uma RSSF

A organização de uma RSSF apresenta características exclusivas deste tipo de rede. O uso de protocolos e algoritmos *genéricos* desenvolvidos para outras redes *ad-hoc* sem fio, na maioria das vezes, não permite explorar as particularidades das RSSFs. Segundo Perkins [2001], RSSFs diferenciam-se de outras redes *ad-hoc* nos seguintes aspectos principais:

- RSSFs são normalmente densas;

- Nós sensores são fortemente limitados em termos de energia disponível, capacidade de processamento e memória;

- A topologia de RSSFs pode mudar frequentemente, tanto no sentido da mobilidade de seus elementos quanto na influência exercida pelo ambiente operacional, que em muitos casos podem ser terrenos inóspitos;

- Nós sensores utilizam principalmente radiodifusão (*broadcasting*) para envio de mensagens, enquanto redes *ad-hoc* são comumente baseadas em comunicação ponto-a-ponto.

Devido ao tamanho limitado e à baixa capacidade dos nós sensores, o controle do consumo de energia talvez seja a mais importante restrição na operação de uma RSSF [Wang et al., 2005a]. Em grande parte das aplicações, os nós sensores não podem ter suas baterias recarregadas ou substituídas (por exemplo, em uma RSSF espalhada por uma floresta densa). Enquanto que em redes *ad-hoc* tradicionais procura-se obter um alto índice de QoS, protocolos de RSSFs precisam se destinar fundamentalmente a reduzir o consumo de energia em sua operação. Normalmente, há um compromisso entre o prolongamento da vida útil da rede e outras métricas importantes, como o atraso médio na entrega de mensagens (o tempo médio entre o momento em que a mensagem é gerada e o momento em que ela alcança o sorvedouro).

Por outro lado, o aumento no tempo de vida útil da rede tem impacto positivo na taxa de cobertura (a porcentagem total da área que é coberta por pelo menos um nó sensor em um determinado instante de tempo). Se os nós sensores conseguem preservar sua energia por um período maior, naturalmente pode-se verificar uma maior taxa de cobertura, principalmente em períodos de tempo mais avançados.

Por estes motivos, várias técnicas têm sido desenvolvidas com o objetivo de reduzir o consumo de energia e, consequentemente, aumentar o tempo de vida útil da rede. Dentre estas técnicas, podemos citar estratégias para disseminação mais eficiente das informações na rede e controle de densidade, explicado a seguir.

2.2.1 Controle de Densidade

A alta densidade em RSSFs é justificada, pelo menos parcialmente, pelo fato de que a autonomia energética dos nós é bastante limitada. Para efeitos de modelagem, toda a área a ser sensoriada é dividida em um conjunto finito de pontos de demanda. Em RSSFs, é comum a existência de muitos nós sensores para cobrir uma determinada área, à qual associa-se demandas de sensoriamento. Em redes muito densas, cada ponto de demanda é usualmente coberto por vários nós sensores, caracterizando assim redundância no grau de cobertura.

Este fato sugere que, ao longo da vida útil da rede e principalmente no início dela, muitos nós poderiam ser desligados sem implicar que algum ponto de demanda fique descoberto. Esta estratégia, denominada Controle de Densidade, é bastante explorada na literatura [Slijepcevic e Potkonjak, 2001; Zhang e Hou, 2005; Siqueira et al., 2006; Aioffi, 2007]. O controle de densidade tem como objetivo atribuir, em um determinado

período de tempo, cada ponto de demanda a um ou mais nós sensores, de modo a garantir a máxima cobertura possível da rede com o mínimo número de nós sensores ativos.

Utilizando-se desta estratégia, pode ser possível aumentar a vida útil da rede ao manter ativo um subconjunto pequeno de nós por um período de tempo determinado, enquanto os demais são mantidos inativos (com seus rádios de comunicação desligados). O problema de definir tal subconjunto de nós que devem ficar ativos com o objetivo de minimizar o consumo de energia é conhecido como o Problema do Controle de Densidade [Nakamura et al., 2005] (PCD). Em conjunto com o uso de outras estratégias, prover a rede com algoritmos de controle de densidade permite elevadas reduções no consumo de energia.

Na literatura, observa-se pelo menos dois grupos principais de trabalhos que investigam como o Controle de Densidade deve ser implementado: aqueles que utilizam abordagens centralizadas, onde a decisão é tomada por uma entidade (normalmente o sorvedouro), e aqueles que utilizam abordagens descentralizadas, onde os próprios nós sensores decidem, de forma colaborativa, qual será o subconjunto que permanecerá ativo.

Um trabalho que explora a abordagem centralizada foi desenvolvido por Slijepcevic e Potkonjak [2001]. Nele, são introduzidas heurísticas que selecionam conjuntos disjuntos de nós, cujos integrantes são capazes de cobrir completamente a área sensoriada. Em cada período de tempo, mantêm-se ativo apenas um desses conjuntos. Porém, para utilizar este procedimento é necessário garantir que a rede seja densa o suficiente para que possam ser encontradas partições do conjunto de nós sensores que exibam as propriedades mencionadas.

Esta restrição não é imposta para a aplicação da abordagem de Zhang e Hou [2005], onde uma estratégia descentralizada é utilizada. A ideia básica do algoritmo proposto naquela referência, denominado OGDC (*Optimal Geographical Density Control*), é que o próprio nó sensor decida se deve ou não permanecer ativo em um determinado período de tempo. Isto é feito através da comunicação com outros nós que cobrem os mesmos pontos de demanda, da seguinte forma. Periodicamente, os nós verificam se podem contribuir com o aumento na cobertura da rede. Aqueles que decidirem que podem contribuir (através de mensagens trocadas com os nós vizinhos) permanecerão ativos por um período determinado de tempo. Após este período, novamente, os nós partem para o processo de decisão. Para que este método funcione corretamente, é necessário que todos os nós sensores possuam relógios sincronizados. Talvez esta seja a maior restrição ao uso desta abordagem.

No trabalho de Siqueira et al. [2006], o algoritmo OGDC foi integrado ao problema

de roteamento de mensagens em redes com sorvedouro fixo. Os nós que estão desativados em um determinado período de tempo não são considerados para o roteamento. Com esta abordagem, foi possível aumentar a taxa de mensagens coletadas pelos nós sensores que foram corretamente entregues ao observador da rede.

Os trabalhos que utilizam o algoritmo OGDC obtiveram bons resultados para redes com sorvedouro fixo quando comparados a outras abordagens. Entretanto, a abordagem distribuída exige trocas de mensagens entre nós sensores em seus processos de decisão, o que consome energia. Além disso, uma abordagem distribuída pode ser menos eficiente que uma abordagem centralizada, no sentido em que os processos de decisão são baseados em conhecimentos locais sobre a rede, enquanto que em uma abordagem centralizada, há conhecimento global.

Em redes que utilizam sorvedouro fixo, a abordagem centralizada potencialmente consome mais energia que a abordagem distribuída, uma vez que mais trocas de mensagens entre nós sensores são necessárias para a disseminação das ordens do controle de densidade. Porém, em redes que utilizam sorvedouros móveis, a abordagem centralizada não necessariamente exige trocas de mensagens entre nós sensores. Caso os sorvedouros sejam móveis e tenham conhecimento das posições e estado de energia dos nós sensores, a decisão pode ser centralizada, baseada em conhecimento global sobre a rede, e comunicada pelo próprio sorvedouro a cada nó sensor. Assim, apenas uma troca de mensagens sorvedouro / nó sensor é necessária para transmitir o novo estado do nó (ativado ou desativado) no próximo período de tempo. Esta abordagem é explorada por Aioffi [2007], onde escolhe-se periodicamente um conjunto de nós sensores com maior energia residual disponível, que cubra a máxima área possível da rede. Este conjunto é ativado enquanto o sorvedouro caminha pela rede. A cada ciclo (volta completa do sorvedouro), um novo subconjunto de nós é calculado.

Como pode ser observado na implementação de estratégias de Controle de Densidade, a forma como a informação de energia da rede é compartilhada é de fundamental importância. A seguir, apresentamos como a disseminação de informações entre os constituintes da rede afeta, de modo geral, os parâmetros de desempenho de RSSFs.

2.2.2 Disseminação da Informação

Segundo Kim et al. [2003], a definição da topologia de uma RSSF envolve, além da densidade dos nós sensores, estabelecer como a informação será disseminada entre os nós e o sorvedouro. Dentre as três funções primárias de um nó sensor (sensoriamento, comunicação e processamento), a comunicação é onde consome-se mais energia [Akyildiz et al., 2002; Kim et al., 2003].

Uma propriedade importante na definição da estratégia de disseminação de infor-

mação em RSSFs é o número de saltos no caminho da transmissão de dados entre o nó sensor e o sorvedouro [Al-Karaki e Kamal, 2004]. Nesse sentido, RSSFs podem ser classificadas como *single-hop*, quando a retransmissão de mensagens entre nós sensores não é permitida, e *multi-hop*.

Normalmente, a comunicação *single-hop* não é uma alternativa viável em redes cujos elementos são fixos, em especial o sorvedouro. Isto ocorre porque o raio limitado dos nós sensores não permite que haja este tipo de comunicação em redes onde os elementos estão separados por longas distâncias. Desta forma, nós sensores têm que transmitir mensagens para outros nós sensores até que a informação chegue ao sorvedouro. Nestes casos, os nós sensores cuja localização é próxima ao sorvedouro tendem a esgotar suas baterias mais rapidamente [Luo e Hubaux, 2005], por serem utilizados mais vezes para rotear mensagens entre o sorvedouro e os nós sensores mais distantes.

A necessidade da comunicação *multi-hop* e seu consumo mais elevado de energia motivaram diversos trabalhos na literatura a procurar estabelecer arquiteturas de RSSFs que utilizam sorvedouro fixo onde o consumo de energia é minimizado. Os trabalhos variam quanto às estruturas de comunicação apresentadas, desde estruturas simples, como árvores ou um conjunto de estrelas conectadas, até estruturas mais complexas.

Um exemplo destas estruturas é aquela proposta por Heinzelman et al. [2002]: o protocolo LEACH (*Low-Energy Adaptive Clustering Hierarchy*), onde *clusters* distribuídos são definidos de forma a permitir a auto-organização dos nós sensores. Para cada *cluster*, há um *cluster head*, um nó sensor que contém equipamento diferenciado, responsável por fazer a comunicação entre os nós e o restante da rede. O protocolo é dotado de algoritmos que adaptam os *clusters* e a posição dos *cluster heads* para distribuir homogeneamente a carga de energia entre todos os nós. Porém, a desvantagem desta abordagem é a necessidade de um grupo de nós sensores com maior capacidade de energia e processamento.

Nós sensores com equipamento diferenciado aumentam o custo da rede e não representam alternativa viável em muitas aplicações. Nestes casos, outros tipos de políticas de disseminação de informação são necessárias, como a proposta por Machado et al. [2005]. Naquele trabalho, a trajetória para a transmissão de mensagens entre um determinado nó sensor e o sorvedouro fixo é definida com base no mapa de energia da rede (disponibilidade de energia dos nós da rede). As rotas da informação são determinadas dinamicamente de acordo com o nível de energia dos nós sensores em um determinado instante de tempo. A rede apresenta capacidade de adaptar seu comportamento conforme os recursos disponíveis. Porém, como apontado anteriormente, os nós mais próximos do sorvedouro tendem a ser mais utilizados por participarem de muitas rotas entre o sorvedouro e os nós mais distantes. Este fato aumenta o risco da perda da

conectividade da rede quando estes nós descarregarem suas baterias.

Uma forma de diminuir tal risco é direcionar o funcionamento da rede a eventos, principalmente quando o comportamento da rede varia com alta frequência. Figueiredo et al. [2004] propuseram o protocolo MULTI, que incorpora vários algoritmos de disseminação de dados que exploram este tipo de alternativa. Em redes com alta variabilidade na geração de informação a ser sensoriada, longos períodos sem incidência de eventos podem ocorrer, mas em determinado momento pode haver um alto tráfego de dados (i.e. um incêndio). A proposta do MULTI consiste em adaptar o seu funcionamento de forma autônoma, adotando o algoritmo mais interessante sob a ótica do consumo de recursos da rede para cada situação. Este protocolo permite a incorporação de diversos mecanismos presentes na literatura como resolvedores internos para cada tipo de situação.

Apesar de todos os esforços de pesquisas citados, a utilização de sorvedouros fixos ainda é uma alternativa de projeto que implica em alto consumo de energia. Assim, é natural considerar a mobilidade do sorvedouro como alternativa. Discutimos este aspecto a seguir.

2.2.3 Mobilidade do Sorvedouro

A mobilidade do sorvedouro não apenas é uma forma de reduzir o gasto de energia e estender o tempo de vida, como também permite que redes esparsas sejam conectadas. Infelizmente, estes ganhos vêm acompanhados da deterioração de algumas medidas de QoS importantes da rede.

Uma vez que a velocidade do sorvedouro é muito menor que a velocidade de transmissão de informação entre nós sensores, permitir que o sorvedouro se mova pela rede coletando mensagens aumenta substancialmente o atraso médio na entrega de mensagens. O impacto deste indesejado efeito pode ser diminuído, pelo menos parcialmente, tanto pelo uso de múltiplos sorvedouros (veja [Somasundara et al., 2007] e [Wang et al., 2005b]), como pelo desenvolvimento de algoritmos de roteamento apropriados [Aioffi, 2007].

A mobilidade do sorvedouro tem sido bastante explorada na literatura, ainda que não tanto quanto o desenvolvimento de estruturas de comunicação para redes com sorvedouro fixo. Como esperado, o objetivo da maior parte destes trabalhos é aumentar o tempo de vida útil da rede, ao substituir a comunicação entre nós sensores pela comunicação sorvedouro / sensor. Pesquisas nesta linha diferem na forma como o movimento do sorvedouro é gerenciado, podendo ser controlado ou não.

Como exemplo de um sorvedouro com movimento não controlado, podemos pensar em um pequeno elemento preso a um animal em movimento ou atrelado a um veículo

de movimentação livre. Os trabalhos de Shah et al. [2003] e Jain et al. [2006] exploram estas ideias, onde o movimento do sorvedouro, além de não controlado, é não previsível. Por estes motivos, nestas abordagens não há como prever o tempo máximo da entrega de mensagens.

Uma forma de lidar com esta dificuldade é apresentada por Chakrabarti et al. [2003], onde sorvedouros são presos a veículos de transporte público, como ônibus e trens, que possuem rotas pré-definidas. Nestes casos, o movimento do sorvedouro, apesar de não controlado, é previsível. Tal abordagem permite alcançar ganhos significativos em economia de energia quando comparado a redes com sorvedouros fixos. Porém, mais uma vez, apresenta algumas importantes restrições, tais como a possível presença de nós sensores distantes das rotas pré-definidas e o fato de que estas mesmas rotas podem ser muito longas, acarretando em altas taxas de atraso na entrega de mensagens.

Figura 2.2. Modelo de sorvedouro móvel Khepera-III

Por estas razões, a maior parte dos estudos que procuram prover mobilidade ao sorvedouro também procuram dotá-lo de controle sobre sua trajetória. Assim sendo, podemos imaginar o sorvedouro como um pequeno robô. Como exemplo, podemos citar o modelo *Khepera-III* (ver Figura 2.2) desenvolvido pela empresa KTeam [Khepera-III, 2009], comercialmente disponível.

Dentre os trabalhos que utilizam sorvedouros com movimento controlado, podemos citar o modelo proposto por Gandham et al. [2003], onde várias bases móveis são utilizadas. O horizonte de tempo é dividido em *rodadas*. Periodicamente, um problema formulado como um Programa Inteiro Misto é resolvido para decidir a localização das bases móveis a cada *rodada*. Este modelo apresenta uma série de desvantagens. Uma delas é que o tempo disponível para a resolução do Programa nem sempre permite que o valor ótimo seja alcançado, mesmo nas pequenas instâncias testadas (redes com

até 30 nós). Outra desvantagem é que não há garantia da eliminação da comunicação *multi-hop*. Isto ocorre porque, a cada rodada, todos os nós sensores devem se comunicar com alguma base móvel, mesmo que, na solução encontrada, estejam distantes da base mais próxima. Desta forma, os nós distantes devem empregar outros nós sensores como pontos intermediários para a comunicação com a base móvel.

A comunicação *multi-hop* também é uma alternativa considerada nos trabalhos de Wang et al. [2005a] e Jea et al. [2005]. No primeiro, é utilizado um sorvedouro móvel que move-se pelo perímetro da área a ser sensoriada. O modelo procura redirecionar as mensagens na direção da extremidade mais próxima; mesmo assim, os nós mais centrais, distantes das extremidades da área sensoriada, comunicam-se com o sorvedouro móvel através de nós intermediários. Já no segundo trabalho, são utilizados vários sorvedouros móveis que se movimentam em linhas retas paralelas. O problema de roteamento se resume em controlar a velocidade de movimentação dos sorvedouros. Caso haja nós sensores que não estejam no raio de comunicação da rota dos sorvedouros, é efetuada a comunicação *multi-hop* para que toda a rede seja coberta.

Os trabalhos de Gandham et al. [2003], Wang et al. [2005a] e Jea et al. [2005] permitem comunicação *multi-hop*. Apesar disto, por dotar o sorvedouro de mobilidade, ganhos consideráveis em termos de consumo de energia foram obtidos quando comparados a redes com sorvedouros fixos. Porém, o tempo gasto nas rotas dos sorvedouros impacta negativamente o atraso na entrega de mensagens. As baixas velocidades de movimentação dos sorvedouros não devem ser negligenciadas, ao contrário do que é assumido no trabalho de Wang et al. [2005b].

Naquele trabalho, são utilizados vários sorvedouros móveis com o objetivo de maximizar o tempo de vida da rede. Um modelo de Programação Linear é formulado para resolver o problema combinado de determinar o movimento do sorvedouro e seu tempo de permanência em diferentes pontos da rede. Para redes de até 256 nós, o modelo proposto alcança tempos de vida da rede até cinco vezes maiores que os obtidos em uma rede totalmente estática. Contudo, o modelo apresenta apenas o tempo necessário que os sorvedouros devem ficar próximos de cada nó para coletar e transmitir as mensagens. Segundo os autores, o tempo de viagem entre os pontos de parada dos sorvedouros é *desprezível*. Na prática, esta informação não condiz com a realidade.

2.3 A Nossa Contribuição

Como pode ser verificado nas Seções anteriores, problemas de otimização complexos com objetivos conflitantes são abundantes em RSSFs. Desta forma, uma RSSF necessita incorporar mecanismos que permitam balancear o tempo de vida útil da rede e

requisitos específicos de QoS. A otimização de parâmetros de QoS em RSSFs, por sua vez, deve sempre ser conduzida levando-se em consideração o impacto das mesmas no tempo de vida da rede.

Neste trabalho, introduzimos algoritmos de otimização que permitem melhorar parâmetros de QoS em RSSFs, tais como o atraso na entrega de mensagens e a taxa de cobertura. A rede considerada aqui envolve múltiplos sorvedouros móveis e centenas de nós sensores aleatoriamente distribuídos. A área de sensoriamento, modelada por um largo quadrado no plano Euclideano, abrange conjuntos discretizados de pontos de demanda, cada um com exigências de sensoriamento uniformes.

Na rede considerada neste estudo, a comunicação só pode ocorrer entre nós sensores e sorvedouros. Sendo assim, o problema que procuramos resolver é o de encontrar um conjunto de boas rotas, uma para cada sorvedouro móvel, que permitam aos sorvedouros coletar a informação sensoriada em toda a rede. Ao invés de visitar cada nó sensor, apenas um subconjunto deles, chamados *cluster heads*, são visitados. Apenas quando o sorvedouro chega a um *cluster head*, a comunicação entre o sorvedouro e todos os nós sensores atribuídos àquele *cluster head* é efetuada.

Ao invés de definir quais nós sensores serão visitados (por exemplo, ao resolver um problema de clusterização) e só então definir as rotas dos sorvedouros (resolvendo o problema do roteamento), abordamos ambos os problemas de forma integrada. Isto é realizado pela forma como as rotas são criadas nos algoritmos e modelos propostos, impondo que cada nó sensor seja um *cluster head* de alguma rota ou que esteja *suficientemente próximo* de algum *cluster head* em alguma rota.

O problema de encontrar, de forma integrada, um conjunto de *cluster heads* e um conjunto de rotas entre eles (uma rota por sorvedouro) é aqui denominado de Problema Integrado de Roteamento e Clusterização (PIRC). Com o objetivo de obter baixos níveis de atraso médio na entrega de mensagens, modelamos o PIRC como uma versão não capacitada do Problema de Roteamento de Veículos [Dantzig e Hamser, 1959], onde o tamanho da frota é previamente conhecido, o objetivo é minimizar o comprimento da maior rota e onde nem todos os clientes (sensores) precisam ser visitados.

Considerando os bons resultados obtidos pelos algoritmos de otimização (especialmente as metaheurísticas) introduzidos neste trabalho, um arcabouço de simulação que integra a resolução do PIRC com mecanismos de controle de densidade foi implementado e testado computacionalmente. Os resultados de otimização e simulação indicam que os algoritmos propostos permitiram a obtenção de reduções significativas nas taxas de atraso médio na entrega de mensagens. Embora menos eficiente que outras estratégias da literatura para a redução do gasto global de energia, o modelo desenvolvido também foi capaz de prover uma maior taxa de cobertura e tempo de vida útil da rede,

graças a execução mais frequente das políticas de controle de densidade. O modelo proposto que permitiu tais ganhos é apresentado no capítulo que segue.

Capítulo 3

Um Modelo Integrado Para o Roteamento e Clusterização em RSSFs

Neste capítulo, ao propor um modelo integrado para simultaneamente tratar as questões de clusterização e roteamento em RSSFs, introduzimos um novo Problema de Otimização Combinatória. Trata-se de uma variante do Problema de Roteamento de Veículos que, a julgar pela nossa revisão bibliográfica, ainda não foi estudado. O problema em estudo, denominado *Problema Integrado de Roteamento e Clusterização* (PIRC), é formulado através de um Problema de Otimização em Grafos. Duas formulações de Programação Inteira são também apresentadas.

3.1 Motivação

Em um estudo recente, Aioffi [2007] propôs o método *Single Hop Strategy* (SHS) para estabelecer um modelo para a disseminação, a recepção e a transmissão de dados em uma RSSF. No método SHS, o sorvedouro comunica-se diretamente com todos os nós da rede; a comunicação entre nós sensores não é permitida. Um único sorvedouro móvel é utilizado para coletar a informação sensoriada. Naquele modelo, assume-se também que a posição geográfica de todos os nós sensores na rede é conhecida *a priori* e que a distância máxima de comunicação entre o sorvedouro e os nós sensores é limitada por um raio $R \geq 0$, um parâmetro que depende do equipamento de rádio utilizado.

O método SHS incorpora algoritmos para resolver os problemas de controle de densidade e roteamento em uma RSSF. Em relação ao roteamento, é utilizada uma abordagem de duas fases. Uma vez que todos os nós sensores necessitam se comunicar com o sorvedouro, eles são divididos em *clusters*. Cada um deles engloba nós sensores que se comunicam com o sorvedouro na medida em que este chega ao centro geométrico do *cluster*. Assim sendo, na primeira fase do método SHS, a rede é dividida em um

número mínimo de *clusters* circulares de raio R. No trabalho de Aioffi [2007], o problema de definir um conjunto minimal de *clusters* é modelado através do Problema dos p-Centros Invertido (PpCI) [Mirchandani e Francis, 1990]. Para resolver o modelo de Programação Inteira associado, o autor empregou pacotes comerciais de otimização. Para as dimensões das redes consideradas naquele estudo, a abordagem de solução empregada para a resolução do PpCI foi considerada satisfatória quanto aos tempos de execução empregados.

Assim que os *clusters* são definidos, tem início a segunda fase do algoritmo, que consiste em determinar o menor circuito Hamiltoniano que passe pelos centros geométricos de todos os *clusters*. Isto é, na segunda fase, resolve-se o Problema do Caixeiro Viajante (PCV) [Dantzig et al., 1954] tendo como conjunto de vértices os centros geométricos dos *clusters*. Para resolver o PCV associado, foi utilizado o Algoritmo de Inserção do Vizinho mais Próximo [Julstrom, 1999].

Ao resolver os problemas de clusterização e roteamento, garante-se a comunicação do sorvedouro com todos os nós da rede. À medida que o sorvedouro percorre a rota, a comunicação entre o sorvedouro e os nós sensores ocorre.

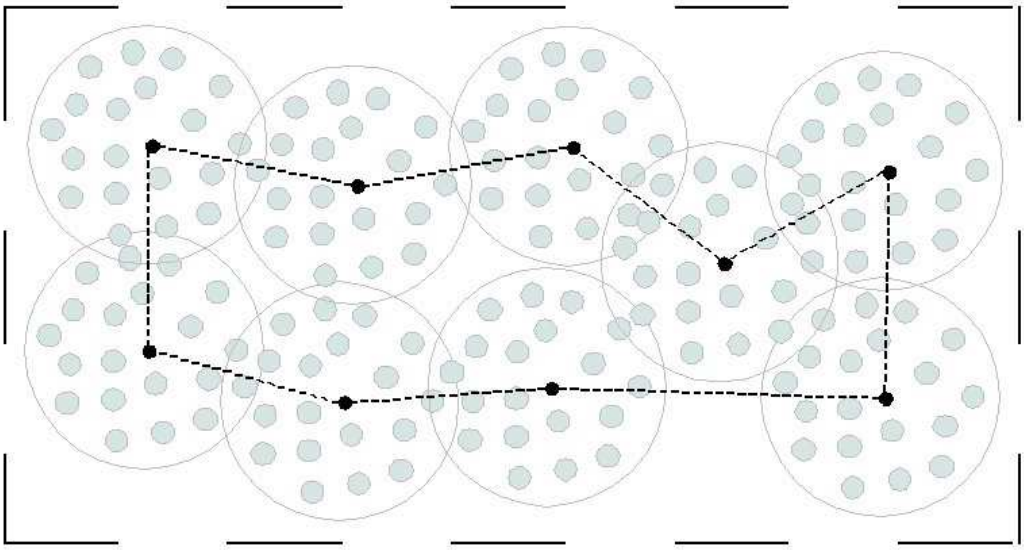

Figura 3.1. Visão geral de uma rota do sorvedouro no método SHS

A Figura 3.1 ilustra como a rede é organizada no método SHS. Na Figura, cada *cluster* é representado por um círculo. O circuito Hamiltoniano que conecta os centros geométricos de cada *cluster* representa a trajetória do sorvedouro pela rede. Apenas quando o sorvedouro chega a um determinado centro de um *cluster* (o qual, neste caso, representa um *cluster head*), a comunicação entre o sorvedouro e todos os nós sensores cobertos por aquele *cluster* é efetuada.

Como a Figura sugere, a vantagem de utilizar uma estratégia de comunicação baseada em *clusters* é permitir que o sorvedouro percorra uma rota menor, visitando apenas o centro de cada *cluster*, ao invés de visitar cada nó sensor. Porém, os problemas de clusterização e roteamento são resolvidos independentemente. Como consequência, um número mínimo de *clusters* não necessariamente implica em rotas de comprimento mínimo. Em tese, ganhos adicionais em termos de atraso na entrega de mensagem poderiam ser obtidos, por exemplo, ao se resolver os problemas de roteamento e clusterização simultaneamente, como proposto nesta dissertação.

3.2 O Modelo Proposto

O Problema Integrado de Roteamento e Clusterização (PIRC) que propomos aqui utiliza-se de padrões de comunicação similares àqueles empregados no SHS. Comunicação direta entre os nós sensores não é permitida; apenas comunicação *single-hop* entre os sorvedouros e os nós sensores pode ocorrer.

O PIRC pode ser descrito da seguinte forma. Dados um conjunto $V = \{1, \ldots, n\}$ de nós sensores (ativos ou inativos, nunca desligados) no plano Euclideano e um conjunto $\mathcal{K} = \{1, \ldots, K\}$ de sorvedouros móveis, o problema que pretendemos resolver consiste em encontrar $K \in \mathbb{Z}_+$ rotas, uma para cada sorvedouro móvel. Cada rota deve incluir alguns nós sensores, denominados *cluster heads*, de forma que todo nó sensor da rede esteja incluído em uma rota (o nó é um *cluster head*) ou esteja a uma distância menor que R de um *cluster head* pertencente a uma das K rotas.

Note que o termo *cluster head* assume agora um significado um pouco diferente. No trabalho de Aioffi [2007], ele foi utilizado para definir os centros geométricos dos *clusters*, os quais eram os locais a serem visitados pelo sorvedouro. No PIRC, o termo define um nó sensor que será visitado por um dos sorvedouros. Apesar de que no PIRC os *cluster heads* são aqueles nós visitados pelos sorvedouros, a eles não é atribuída nenhuma função especial na rede quando comparados aos nós sensores que não são *cluster heads*.

Visando obter baixas taxas de atraso na entrega de mensagens, procuramos encontrar K rotas de forma que o comprimento da maior delas seja minimizado. Na medida em que o valor de K cresce, o atraso médio na entrega de mensagens deve decrescer. Minimizar a rota mais longa permite balancear os comprimentos das K rotas de tal forma que todos os sorvedouros levem aproximadamente o mesmo tempo para coletar a informação dos nós sensores atribuídos à sua rota.

Um importante pressuposto no modelo PIRC é que todos os K sorvedouros iniciam seus movimentos ao mesmo tempo, i.e., eles são sincronizados. Se este não fosse o

caso, a rede poderia ficar desbalanceada, uma vez que um nó sensor visitado por uma rota menor comunicaria mais frequentemente com o sorvedouro a ele atribuído que os nós visitados por rotas mais longas. Outra razão para a sincronização é permitir a implementação de um controle de densidade centralizado, no início de cada ciclo, antes dos sorvedouros iniciarem seus movimentos. Sendo assim, o primeiro sorvedouro que atingir o depósito (ponto inicial e final da rota dos sorvedouros) deve esperar a chegada dos demais para iniciar um novo ciclo de planejamento da rede (que compreende uma travessia completa de todos os sorvedouros pelas suas respectivas rotas).

Para formular o PIRC como um Problema de Otimização em Grafos, utilizaremos um parâmetro R, que define o raio máximo de comunicação entre o sorvedouro e um nó sensor, e um digrafo $D = (V, A)$ com o conjunto de vértices $V = \{1, \ldots, n\}$ e de arcos A. Para este propósito, assuma que, inicialmente, todos os sorvedouros móveis estão localizados em um depósito, representado pelo vértice $1 \in V$. O conjunto de arcos $A := \{(i, j), (j, i) : \forall i, j \in V, i \neq j\}$ representa todas as possíveis translações dos sorvedouros móveis, movendo de um *cluster head* a outro. Um peso $d_{ij} \geq 0$ é atribuído a cada arco $(i, j) \in A$. Neste trabalho, d_{ij} corresponde ao maior inteiro menor ou igual à distância Euclideana entre i e j. Vamos também definir $d_{ii} = 0, \forall i \in V$. Finalmente, considere que $\omega(i) := \{j \in V : d_{ij} \leq R\}$ denota o conjunto de vértices *suficientemente próximos* de i. Observe que diante das definições anteriores $i \in \omega(i)$, $\forall i \in V$.

Uma solução para o PIRC em D é uma coleção de K rotas sujeitas a algumas restrições adicionais. Cada rota $k \in \mathcal{K}$ tem seu início em 1, visita um conjunto $S_k \setminus \{1\}$ de vértices selecionados e retorna ao vértice 1. Referimo-nos ao subgrafo de D induzido por cada rota k como $H_k = (S_k, A_k)$. Consequentemente, $H = \bigcup_{k=1}^{K}(S_k, A_k)$ representa o subgrafo associado ao conjunto completo de K rotas. No que segue, dizemos que $i \notin \bigcup_{k=1}^{K} S_k$ é coberto por j se existe $k \in \mathcal{K}$ tal que $j \in S_k$ e $i \in \omega(j)$. Quando este for o caso, também dizemos que i é coberto pela rota k. Se definirmos $f(H_k) = \sum_{(i,j)\in A_k} d_{ij}$ como o comprimento da $k-$ésima rota, o custo de uma solução viável H para o PIRC é dado por $f(H) = \max\{f(H_k) \ : k = 1, \ldots, K\}$.

Diante do exposto, o PIRC consiste no problema de:

$$\min \ f(H) : H = \bigcup_{k=1}^{K}(S_k, A_k), \tag{3.1}$$

tal que

$$\forall k \in \mathcal{K} : A_k \text{ induz um circuito Hamiltoniano entre os vértices de } S_k, \tag{3.2}$$

$$S_p \cap S_q = \{1\}, \ \forall p, q \in \mathcal{K}, p \neq q, \tag{3.3}$$

$$\forall i \in V : i \in \bigcup_{k=1}^{K} S_k \textbf{ ou } \exists j \in V \setminus \{i\} : \; j \in \bigcup_{k=1}^{K} S_k, \; i \in \omega(j). \tag{3.4}$$

Note que (3.3) garante que o depósito é o único vértice em comum visitado por qualquer par de rotas e que (3.4) impõe que cada vértice é um *cluster head* ou está coberto por algum *cluster head*.

O PIRC é claramente um problema cuja versão de decisão é NP-Completo, uma vez que o Problema do Caixeiro Viajante [Dantzig et al., 1954; Jünger et al., 1995] é um de seus casos especiais, quando $K = 1$ e $\omega(i) = \{i\}, \forall i \in V$ $(R = 0)$.

De acordo com nossa revisão bibliográfica, a variante do Problema de Roteamento de Veículos (PRV) mais próxima do PIRC é aquela discutida por Glaab [2002]. Naquele trabalho, os autores introduzem um Problema de Roteamento de Veículos que surge no contexto do projeto de sistemas semi-automáticos de corte de couro. Assim como no PIRC, deseja-se minimizar o comprimento da rota mais longa e o tamanho da frota é fixo. Entretanto, o PIRC difere daquela variante do PRV em dois aspectos fundamentais: *(i)* por aquela variante não apresentar natureza seletiva (i.e. todos os clientes devem ser visitados) e *(ii)* pelo fato de que cada veículo inicia sua trajetória de um depósito diferente.

Cabe mencionar que outros problemas de Otimização Combinatória guardam similaridades com o PIRC exatamente por exibir uma natureza seletiva. Como exemplos, podemos citar o *Covering Tour Problem* [Gendreau et al., 1997], o Problema do Caixeiro Viajante Seletivo [Gendreau et al., 1998] e o problema do Caixeiro Viajante Generalizado [Fischetti et al., 1997]. Todos estes três problemas são semelhantes ao PIRC no caso especial em que $K = 1$. Entretanto, todos diferem do PIRC de alguma forma.

Dados conjuntos de vértices T, V, W, tais que $T \subseteq V$, no *Covering Tour Problem* deseja-se encontrar um circuito hamiltoniano de custo mínimo que passe por todos os vértices de T. Em adição a estes, podem também ser visitados vértices em $V \setminus T$. O circuito escolhido deve ser tal que todo vértice em W esteja *suficientemente próximo* de algum vértice visitado. Observe que quando $K = 1$, o PIRC difere do *Covering Tour Problem* já que no PIRC não existe um conjunto de vértices terminais T que necessariamente precisa ser visitado.

Assim como o *Covering Tour Problem*, no Problema do Caixeiro Viajante Seletivo há um conjunto de vértices T que deve ser visitado. Além de custos serem atribuídos às arestas do grafo, prêmios não negativos são também associados aos seus vértices. Assim sendo, deseja-se obter um circuito cuja soma dos prêmios dos vértices visitados seja máximo e que a soma dos custos das arestas envolvidas não exceda um orçamento previamente estabelecido.

No Problema do Caixeiro Viajante Generalizado, por sua vez, os vértices do grafo de definição do problema são previamente organizados em *clusters* (conjuntos de vértices disjuntos). O objetivo consiste então em obter um *tour* de mínimo custo que visite pelo menos um vértice de cada *cluster*. Observe que este problema difere do PIRC ($K = 1$) uma vez que neste último, a organização dos vértices em *clusters* não é previamente estabelecida.

Nas Seções seguintes, apresentamos dois modelos de Programação Inteira para o PIRC: o primeiro baseado em Fluxos em Redes e o segundo baseado em Desigualdades de Eliminação de Subrotas.

3.3 Uma Formulação de Fluxos para o PIRC

O primeiro modelo de Programação Inteira que apresentamos para o PIRC é baseado em Fluxos em Redes [Ahuja et al., 1993]. Sua principal ideia é atribuir uma mercadoria $k \in \mathcal{K}$, inicialmente disponível no vértice depósito, a toda rota $k \in \mathcal{K}$. Um vértice deve receber uma única unidade da mercadoria k se e somente se for um *cluster head* na rota k. Neste caso, uma unidade de cada mercadoria k deve então ser entregue do depósito àquele vértice, utilizando arcos apropriados da rede.

Para formularmos o problema, modelaremos cada rota de cada sorvedouro como um caminho simples em um digrafo $\overline{D} = (\{0\} \cup V, \overline{A})$ obtido ao se adicionar a D:

- (i) um vértice artificial 0 (uma cópia do depósito 1) em conjunto com
- (ii) um conjunto de arcos artificiais $\{(i, 0) : d_{i0} = d_{1i},\ \forall i \in V \setminus \{1\}\}$ incidentes a 0.

Como resultado, temos $\overline{A} := A \cup \{(i, 0) : \forall i \in V \setminus \{1\}\}$.

Não é difícil perceber que, por construção, o comprimento de um caminho simples em $\overline{D}$ que começa em 1, visita cada vértice em $S_k \setminus \{1\}$ exatamente uma vez e termina em 0 é precisamente $f(H_k)$.

Para formular o PIRC como um Problema de Fluxos em Redes sujeito a restrições complicantes, os seguintes conjuntos de variáveis de decisão serão empregados:

- $y_i^k \in \mathbb{B}, \forall i = 0, \ldots, n, \forall k \in \mathcal{K}$, assumindo valor 1 se i é um *cluster head* na rota k (0, caso contrário);
- $x_{ij}^k \in \mathbb{B}, \forall (i, j) \in \overline{A}, \forall k \in \mathcal{K}$, assumindo valor 1 se o arco (i, j) é selecionado para pertencer à $k-$ésima rota (0, caso contrário);

- $v_{ij}^k \in \mathbb{R}_+, \forall (i,j) \in \overline{A}, \forall k \in \mathcal{K}$, indicando a quantidade de mercadoria k que flui pelo arco (i,j);
- $w \in \mathbb{R}_+$ denotando o comprimento da mais longa das K rotas.

Uma formulação para o PIRC é dada por:

$$f = \min \ \{w : (w, v, x, y) \in \mathcal{P}_{FLUXO} \cap (\mathbb{R}_+, \mathbb{R}_+^{K|\overline{A}|}, \mathbb{B}^{K|\overline{A}|}, \mathbb{B}^{K|V\cup\{0\}|})\}, \tag{3.5}$$

onde $\mathcal{P}_{FLUXO}$ é o poliedro definido pelas restrições:

$$\sum_{i \in V\setminus\{1\}} v_{1,i}^k = \sum_{i \in \{0\}\cup V\setminus\{1\}} y_i^k, \ \ \forall k \in \mathcal{K}, \tag{3.6}$$

$$\sum_{j \in \{0\}\cup V\setminus\{1\}} v_{ij}^k - \sum_{j \in V} v_{ji}^k = -y_i^k, \ \ \forall i \in V \setminus \{1\}, \ \ \forall k \in \mathcal{K}, \tag{3.7}$$

$$\sum_{i \in V} v_{i,0}^k = 1, \ \ \forall k \in \mathcal{K}, \tag{3.8}$$

$$v_{ij}^k \leq n x_{ij}^k, \ \ \forall (i,j) \in \overline{A}, \ \forall k \in \mathcal{K}, \tag{3.9}$$

$$x_{ij}^k \leq y_i^k, \ \ \forall (i,j) \in \overline{A}, \ \forall k \in \mathcal{K}, \tag{3.10}$$

$$x_{ij}^k \leq y_j^k, \ \ \forall (i,j) \in \overline{A}, \ \forall k \in \mathcal{K}, \tag{3.11}$$

$$\sum_{k \in \mathcal{K}} y_i^k \leq 1, \ \ \forall i \in V \setminus \{1\}, \tag{3.12}$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in \omega(i)} y_j^k \geq 1, \ \ \forall i \in V \setminus \{1\}, \tag{3.13}$$

$$\sum_{j \in V\cup\{0\}} x_{ij}^k \leq 1, \ \ \forall i \in V, \forall k \in \mathcal{K}, \tag{3.14}$$

$$w \geq \sum_{(i,j) \in \overline{A}} d_{ij} x_{ij}^k, \ \ \forall k \in \mathcal{K}, \tag{3.15}$$

$$y_1^k = y_0^k = 1, \ \ \forall k \in \mathcal{K}, \tag{3.16}$$

$$y_i^k \geq 0, \ \forall i \in V \cup \{0\}, \ \forall k \in \mathcal{K}, \tag{3.17}$$

$$v_{ij}^k \geq 0, \ \forall (i,j) \in \overline{A}, \ \forall k \in \mathcal{K}, \tag{3.18}$$

$$x_{ij}^k \geq 0, \ \forall (i,j) \in \overline{A}, \ \forall k \in \mathcal{K}. \tag{3.19}$$

Observe que (3.6)-(3.8) são restrições de conservação de fluxo para os vértices em

$\{1\}$, $V \setminus \{1\}$ e $\{0\}$, respectivamente. Note que as restrições (3.6) impõem que a quantidade de mercadorias do tipo k que deixam o depósito é $|(S_k \setminus \{1\}) \cup \{0\}|$. As restrições (3.7), por outro lado, garantem que um vértice i que é visitado pela rota k deve reter uma unidade da mercadoria k. As desigualdades (3.9)-(3.10), por sua vez, são restrições de acoplamento. Elas garantem que só pode haver fluxo da mercadoria k em um arco caso o mesmo seja selecionado para fazer parte da rota k. Adicionalmente elas impõem que um arco é selecionado para estar em uma rota somente se suas extremidades receberem uma unidade da mercadoria correspondente. As restrições (3.12) garantem que nenhum *cluster head* será visitado por mais de uma rota. As desigualdades (3.13) asseguram que cada nó sensor é um *cluster head* ou é coberto por um *cluster head*. Em conjunto, as desigualdades (3.6)-(3.8) e (3.14) garantem que a topologia dos arcos selecionados induzem K caminhos simples conectando 1 e 0. Finalmente, as restrições (3.15) são utilizadas para definir o maior comprimento de rota a ser minimizado em (3.5).

A formulação $\mathcal{P}_{FLUXO}$ é dita compacta por possuir um número de restrições e variáveis que cresce polinomialmente na medida em que o número de vértices aumenta. Esta formulação emprega uma única mercadoria para cada rota para definir o conjunto de vértices nela visitado. É sabido ([Magnanti e Wolsey, 1995]) que formulações de fluxos que empregam múltiplas mercadorias usualmente fornecem limites de Relaxação Linear mais fortes, por permitir estabelecer restrições de acoplamento (do tipo das restrições (3.9)) mais apertadas. Apesar disto, optamos pela formulação apresentada por envolver um menor número de variáveis, uma vez que o conjunto de arcos A é completo.

3.4 Uma Formulação Baseada em Desigualdades de Eliminação de Subrotas

Para apresentarmos uma formulação para o PIRC baseada em Desigualdades de Eliminação de Subrotas Generalizadas (*GSEC*) [Gendreau et al., 1997], empregaremos um grafo não-direcionado $G = (V, E)$ com o conjunto de vértices V (o mesmo utilizado para o digrafo D) e de arestas E. Uma vez que a matriz de distâncias é simétrica, empregaremos aqui um grafo não orientado para formular o PIRC através de um modelo que usa desigualdades *GSEC*. Assim sendo, $E = \{[i, j] : i < j\}$ denota o conjunto completo de arestas cujas extremidades são vértices de V. Na formulação que segue, empregaremos as seguintes definições. Para qualquer $W \subset V$, $E[W, V \setminus W] := \{[i, j] \in E : i \in W, j \in V \setminus W\}$ define o conjunto de arestas no corte $[W, V \setminus W]$ e $E(W) := \{[i, j] \in E : i, j \in W\}$ define o conjunto de arestas com

ambas as extremidades em W.

A formulação baseada em desigualdades *GSEC* emprega os seguintes conjuntos de variáveis:

- $y_i^k \in \mathbb{B}, \forall i = 0, \ldots, n, \forall k \in \mathcal{K}$, assumindo valor 1 se i é um *cluster head* na rota k (0, caso contrário);
- $x_{ij}^k \in \mathbb{B}, \forall [i,j] \in E, \forall k \in \mathcal{K}$, indicando se a aresta $[i,j]$ faz parte da rota k (0, caso contrário);
- $w \in \mathbb{R}_+$ denotando o comprimento da mais longa das K rotas.

Uma formulação baseada em desigualdades *GSEC* para o PIRC é dada por:

$$f = \min \{w : (w,x,y) \in \mathcal{P}_{GSEC} \cap (\mathbb{R}_+, \mathbb{B}^{K|E|}, \mathbb{B}^{K|V|})\}, \tag{3.20}$$

onde $\mathcal{P}_{GSEC}$ é o poliedro definido pelas restrições:

$$\sum_{[i,j] \in E[\{i\}, V \setminus \{i\}]} x_{ij}^k = 2y_i^k, \ \ \forall i \in V, \ \ \forall k \in \mathcal{K}, \tag{3.21}$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in \omega(i)} y_j^k \geq 1, \ \ \forall i \in V, \tag{3.22}$$

$$\sum_{k \in \mathcal{K}} y_i^k \leq 1, \ \ \forall i \in V \setminus \{1\}, \tag{3.23}$$

$$w \geq \sum_{[i,j] \in E} d_{ij} x_{ij}^k, \ \ \forall k \in \mathcal{K}, \tag{3.24}$$

$$\sum_{[i,j] \in E[W, V \setminus W]} x_{ij}^k \geq 2y_z^k, \forall W \subset V, 1 \in W, z \notin W, \forall k \in \mathcal{K}, \tag{3.25}$$

$$y_1^k = 1, \ \ \forall k \in \mathcal{K}, \tag{3.26}$$

$$y_i^k \geq 0, \ \ \forall i \in V, \ \ \forall k \in \mathcal{K}, \tag{3.27}$$

$$x_{ij}^k \geq 0, \ \ \forall [i,j] \in E, \ \ \forall k \in \mathcal{K}. \tag{3.28}$$

Observe que as restrições (3.21) asseguram que sempre que i é visitado pelo $k-$ésimo sorvedouro, exatamente duas arestas devem ser incidentes a i na $k-$ésima rota. As desigualdades (3.22), por sua vez, garantem que cada vértice é um *cluster head* ou está

suficientemente próximo de algum *cluster head* visitado por alguma rota. As restrições (3.23) impõem que um vértice não pode ser visitado por dois ou mais veículos. As restrições (3.24) permitem a minimização da rota mais longa. Finalmente, as Desigualdades de Eliminação de Subrotas Generalizadas (*GSECs*) (3.25) evitam circuitos que não incluam o depósito. Como pode ser verificado, este modelo emprega exponencialmente muitas restrições do tipo (3.25).

3.5 Tratamento da Multiplicidade de Soluções Idênticas Decorrentes da Indexação de Rotas

Em virtude da natureza min/max da função objetivo do PIRC, nas formulações que apresentamos, utilizamos um índice para cada rota. Isto é necessário para podermos capturar o comprimento de cada uma delas e então minimizarmos a rota mais longa. Este artifício de modelagem faz com que duas soluções idênticas nos grafos de definição do problema (seja em $\overline{D}$ ou E) possam corresponder a pontos distintos dos poliedros que definem as formulações empregadas, $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$. A Figura 3.2 ilustra este caso, onde três soluções para uma instância hipotética são apresentadas. Considere que, na Solução A da Figura, a rota com índice $k = 1$ é aquela que possui o maior comprimento, seguida pela rota com índice $k = 2$. A rota com índice $k = 3$, por sua vez, é a mais curta dentre as três rotas. Observe que as Soluções A, B e C diferem entre si apenas por possuírem atribuições distintas de índices para cada rota.

Para o caso das formulações $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$, esta multiplicidade de soluções idênticas, que diferem apenas por diferentes atribuições de índices a cada uma de suas rotas, acarreta complicações adicionais à resolução do PIRC. Isto ocorre porque os algoritmos Branch-and-Bound baseados em $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$ podem ter seu desempenho comprometido por esta mutiplicidade de soluções idênticas. Na prática, há a tendência de que tais algoritmos investiguem ramos "idênticos"da árvore de enumeração.

Uma forma que encontramos para contornar esta dificuldade e melhorar o desempenho dos algoritmos de solução baseados em $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$ é a seguinte. Ordenamos os índices das rotas por comprimento, impondo que a k-ésima rota não terá comprimento menor que a $(k+1)$-ésima rota. Esta observação resulta nas restrições:

$$\sum_{(i,j)\in\overline{A}} d_{ij}x_{ij}^{k} \geq \sum_{(i,j)\in\overline{A}} d_{ij}x_{ij}^{k+1}, \ \ \forall k \in \{1,\ldots,K-1\} \tag{3.29}$$

e

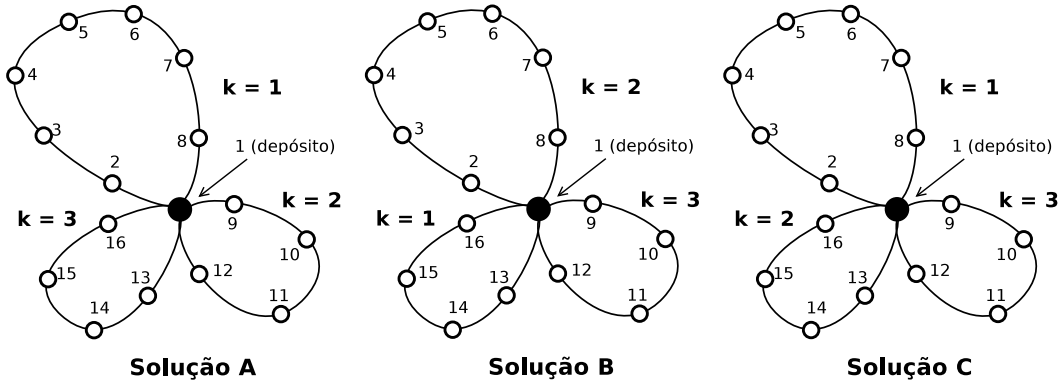

Figura 3.2. Soluções graficamente idênticas, mas com índices diferentes.

$$\sum_{[i,j]\in E} d_{ij}x_{ij}^{k} \geq \sum_{[i,j]\in E} d_{ij}x_{ij}^{k+1}, \ \ \forall k \in \{1,\ldots,K-1\}, \tag{3.30}$$

que podem ser respectivamente acrescentadas às formulações $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$.

Observe que, considerando as soluções apresentadas na Figura 3.2, se acrescentarmos a restrição (3.29) a $\mathcal{P}_{FLUXO}$ e (3.30) a $\mathcal{P}_{GSEC}$, garantimos que apenas a Solução A pertença ao poliedro resultante. A utilização desta restrição assegura que qualquer troca de índices entre rotas resultará em uma solução inviável, a não ser que as rotas trocadas possuam o mesmo comprimento.

Uma outra forma de reduzir a multiplicidade de soluções viáveis é estabelecer que um vértice será ou visitado ou coberto por uma rota pré-determinada. Isto pode ser obtido, por exemplo, ao impor que determinado vértice será um *cluster head* da primeira rota ou estará *suficientemente próximo* de algum *cluster head* da mesma rota. Para tanto, basta adicionar às formulações $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$ a seguinte restrição:

$$\sum_{j\in\omega(z)} y_j^1 \geq 1, \ \ z \in \arg\min\{|\omega(i)| \ \ \forall i \in V\}, \tag{3.31}$$

A escolha do vértice com o menor grau de comunicação para ser fixado na rota de índice $k = 1$ é puramente uma decisão de implementação; não foram realizados testes para verificar qual o melhor vértice a ser escolhido. Além disso, esta restrição apresenta uma potencial desvantagem em relação à desigualdade (3.30): não é possível fixar mais de um vértice em uma rota. Suponha, considerando a Figura 3.2, que decida-se por fixar o nó de índice 5 na rota de índice $k = 1$. Desta forma, a Solução A é viável e a Solução B é inviável. Em compensação, para este exemplo, a Solução C também é viável, pois apenas as rotas de índice $k = 2$ e $k = 3$ foram trocadas.

Devemos salientar que as duas propostas para lidar com a mutiplicidade de soluções

viáveis (restrições (3.29) e (3.30) e a restrição (3.31)) são excludentes, isto é, não podem ser impostas simultaneamente na mesma formulação. Sendo assim, apenas uma das duas estratégias deve ser escolhida para ser incorporada às formulações $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$. Ao contrário do que esperávamos, constatamos empiricamente que, mesmo com a potencial desvantagem descrita anteriormente, a restrição (3.31) mostrou-se mais eficaz que as restrições (3.29) e (3.30), proporcionando, ao final do tempo limite de execução dos experimentos realizados, *gaps* de dualidade menores.

Outra estratégia que empregamos para melhorar o desempenho prático dos algoritmos Branch-and-Bound que baseiam-se em $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$ foi criar mecanismos artificiais de desempate de soluções que diferem apenas pela atribuição de índices às suas rotas. O mecanismo empregado foi o de associar custos c_i^k às variáveis y_i^k e considerá-los na função objetivo do problema. Assim sendo, a função objetivo com a qual efetivamente trabalhamos é:

$$f = \min\ w + \sum_{k \in \mathcal{K}} \sum_{i \in V} c_i^k y_i^k. \tag{3.32}$$

Para que esta estratégia seja empregada, é necessário escolher custos c_i^k tais que, dadas as soluções ótimas do problema original (com a função objetivo original), pelo menos uma ainda seja ótima para (3.32). Os custos c_i^k devem desempatar apenas as soluções do PIRC que, diante da função objetivo original, possuem custos idênticos. Considerando que as instâncias testes que empregamos para validar os algoritmos exatos aqui propostos possuem distâncias $d_{ij} \in \mathbb{Z}_+$, um conjunto de custos $\{c_i^k\}$ que garante os aspectos acima pode ser obtido da seguinte forma:

- inicialmente, para todo par i, k atribuímos a c_i^k um valor aleatório $0 \leq r < 1$, com distribuição uniforme;
- em seguida, recalculamos c_i^k como

$$c_i^k \leftarrow \frac{1}{10} \frac{c_i^k}{\sum_{j \in V} \sum_{l \in \mathcal{K}} c_j^l} \ \ \forall i \in V, \ \ \forall k \in \mathcal{K}.$$

Observe que diante da estratégia acima a contribuição de $\sum_{i \in V} \sum_{k \in \mathcal{K}} c_i^k y_i^k$ na função objetivo é sempre menor que 1.

Nos resultados dos algoritmos exatos descritos no próximo capítulo, utilizamos a estratégia de associar pequenos custos aleatórios ao conjunto de variáveis de decisão y em adição ao uso da restrição (3.31).

Capítulo 4

Métodos de Solução Exata para o PIRC

Neste capítulo, apresentamos três algoritmos exatos para a resolução do PIRC. O primeiro deles é um algoritmo Branch-and-Bound baseado na formulação $\mathcal{P}_{FLUXO}$. O segundo e terceiro são baseados na formulação $\mathcal{P}_{GSEC}$ e tratam de um algoritmo Branch-and-Cut e um algoritmo Local Branching que utiliza o próprio Branch-and-Cut como resolvedor interno. Os resultados computacionais que apresentamos para os três algoritmos indicam que a natureza min/max da função objetivo e o fato de que nem todos os vértices precisam ser visitados tornam o problema muito difícil de ser resolvido, mesmo para instâncias de pequenas dimensões.

4.1 Um Algoritmo Branch-and-Bound Baseado na Formulação de Fluxos

O primeiro algoritmo testado aqui é um algoritmo Branch-and-Bound (BB) [Land e Doig, 1960] que baseia-se na formulação $\mathcal{P}_{FLUXO}$ apresentada para o PIRC. Trata-se de um algoritmo BB que utiliza todas as funcionalidades oferecidas pelo pacote de otimização CPLEX [ILOG Cplex Solver, 2009] (versão 10.2.0 com configuração padrão) para controle e exploração da árvore de enumeração.

Nossos experimentos computacionais com o algoritmo BB foram conduzidos em 11 instâncias testes geradas a partir de instâncias Euclideanas bidimensionais provenientes da biblioteca TSPLIB [TSPLIB, 2009]. Para cada instância da TSPLIB considerada aqui, uma instância correspondente do PIRC foi gerada ao definir o raio R de tal forma que a densidade de comunicação, dada por $\frac{\sum_{i \in V} |\omega(i) \setminus \{i\}|}{n}$, seja o mais próximo possível de 0.75. Isto significa que cada nó sensor pode se comunicar via rádio com, em

média, 0.75% dos nós sensores da rede. Ao tentar resolver o PIRC com técnicas exatas que funcionam relativamente bem para problemas com estruturas semelhantes, como o PCV e o PRV, observamos que a natureza seletiva do problema é um fator complicante na obtenção de bons resultados. O valor de 0.75 para a densidade de comunicação, embora resulte em redes de certa forma esparsas, foi escolhido por permitir estabelecer um certo grau de competição entre os vértices para participar das rotas sem no entanto tornar o problema difícil o suficiente para que conclusões não pudessem ser extraídas da nossa análise.

É importante mencionar que todos os experimentos computacionais apresentados nesta dissertação foram conduzidos em um computador com processador *AMD Dual Core*, que opera em 1.9 GHz e possui 3Gb de memória RAM. O sistema operacional Linux foi utilizado. Os algoritmos foram implementados em C++, compilados com o g++ com *flags* de otimização ligados.

Para a execução do algoritmo, foi imposto um tempo limite de no máximo 4 horas. Caso este limite de tempo seja alcançado, a execução do algoritmo BB é interrompida e os melhores limites superiores e inferiores são recuperados. Antes da execução do BB, uma solução inicial viável foi fornecida para o CPLEX. Esta solução foi obtida através da aplicação do procedimento heurístico `GRASP-ILS/VND`, descrito em detalhes no Capítulo 5.

A Tabela 4.1 apresenta os resultados para cada caso testado, considerando $K \in \{1, 2, 3\}$. A primeira coluna da Tabela indica as instâncias geométricas da TSPLIB escolhidas para a realização dos experimentos computacionais. O tamanho de cada instância (número de vértices) é indicado em seu nome. Por exemplo, a instância *eil51* é composta por 51 vértices. Nas colunas seguintes, são apresentados os resultados obtidos para, respectivamente, $K = 1, 2$ e 3. Para cada valor de K, quatro colunas são apresentadas: a primeira, intitulada $\overline{f}_{HEU}$, apresenta o valor da solução inicial obtida com as heurísticas do Capítulo 5. A segunda coluna, sob a alcunha $\overline{f}$, representa o custo da melhor solução viável encontrada pelo algoritmo BB. Um valor destacado entre parênteses indica que o BB conseguiu melhorar a solução fornecida pelos procedimentos heurísticos. A terceira coluna, $\underline{f}$, indica o melhor limite dual obtido ao longo da árvore de enumeração e, por fim, a quarta coluna apresenta o *gap* de dualidade ($\frac{\overline{f}-\underline{f}}{\overline{f}} \times 100$) remanescente quando o algoritmo foi interrompido.

O algoritmo BB encontrou grandes dificuldades em resolver o PIRC para as instâncias testadas. Para $K = 1$, por exemplo, nenhuma instância foi resolvida e o *gap* de dualidade médio ao término (interrupção) do algoritmo foi de 17.88%. Na medida em que K aumenta, os valores obtidos são ainda piores. Para $K = 2$, o *gap* médio de dualidade ao fim da execução do BB foi de 34.89%, enquanto que, para $K = 3$, foi de

Instância	$K=1$				$K=2$				$K=3$			
	$\overline{f}_{HEU}$	$\overline{f}$	$\underline{f}$	gap (%)	$\overline{f}_{HEU}$	$\overline{f}$	$\underline{f}$	gap (%)	$\overline{f}_{HEU}$	$\overline{f}$	$\underline{f}$	gap (%)
eil51	369	(364)	357.00	1.92	196	196	165.00	15.82	143	143	111.46	22.06
eil76	479	(464)	449.71	3.08	261	(259)	200.38	22.63	188	188	136.59	27.35
rat99	1067	(1046)	930.54	11.04	610	(594)	421.47	29.05	507	507	290.14	42.77
eil101	617	(616)	569.00	7.63	327	327	274.00	16.21	238	(237)	197.33	16.74
bier127	117241	(114349)	103516.57	9.47	60299	(60187)	45154.68	24.96	44082	44082	30249.18	31.38
ch130	5749	(5535)	4201.53	24.09	2999	2999	1884.02	37.18	2240	2240	1285.23	42.62
pr136	76702	(66116)	50157.95	24.14	43520	43520	21612.52	50.34	33209	33209	14687.91	55.77
pr144	57333	57333	28127.42	50.94	34139	(34068)	9909.61	70.91	28982	28982	6878.90	76.26
ch150	5518	(5299)	4482.20	15.41	3089	3089	2029.78	34.29	2331	2331	1364.30	41.47
rat195	1760	1760	1318.05	25.11	993	(963)	545.57	43.35	778	778	367.22	52.80
tsp225	3170	3170	2413.64	23.86	1780	1780	1085.00	39.04	1391	1391	742.87	46.59
Gap médio				17.88				34.89				41.44

Tabela 4.1. Resultados computacionais - Algoritmo Branch-and-Bound baseado na Formulação de Fluxos

41.44%.

Outro fator que merece ser observado é a dificuldade do BB em melhorar a solução viável inicialmente fornecida, especialmente para $K > 1$. Para $K = 2$, o BB conseguiu melhorar a solução inicial em apenas 5 dos 11 casos, enquanto que para $K = 3$, foi possível obter uma solução de custo menor para apenas um caso.

As dificuldades encontradas com o algoritmo baseado neste modelo apontaram a necessidade de adotar outro tipo de estratégia de solução exata, na tentativa de alcançar melhores resultados. Para tanto, serão propostos, nas próximas Seções, duas outras abordagens: um algoritmo Branch-and-Cut [Grötschel et al., 1984; Padberg e Rinaldi, 1991] e um algoritmo Local Branching [Fischetti e Lodi, 2003].

4.2 Um Algoritmo Branch-and-Cut

O algoritmo Branch-and-Cut (BC) [Grötschel et al., 1984; Padberg e Rinaldi, 1991] consiste em um método que incorpora um algoritmo de Planos de Corte [Dantzig et al., 1954; J. E. Kelley, 1960] a um procedimento enumerativo inteligente, do tipo Branch-and-Bound.

Como dito anteriormente, o poliedro $\mathcal{P}_{GSEC}$ apresenta um número exponencial de restrições. Desta forma, incorporar todas elas e resolver o modelo através de um algoritmo Branch-and-Bound é uma tarefa inviável para instâncias do PIRC já de tamanho relativamente pequeno. Ao invés disto, introduziremos *GSECs* em relaxações lineares para (3.20), na medida em que forem necessárias. Assim sendo, iniciamos o algoritmo Branch-and-Cut resolvendo:

$$\min \{w : (w, x, y) \in \mathcal{P}'\}, \tag{4.1}$$

onde $\mathcal{P}'$ é dado pela interseção de (3.21)-(3.24), (3.26)-(3.28) e as seguintes desigualdades lógicas:

$$\begin{aligned} x_{ij}^k &\leq y_i^k \qquad \forall\, [i,j] \in E, \forall\, k \in \mathcal{K}. \\ x_{ij}^k &\leq y_j^k \end{aligned} \tag{4.2}$$

Assuma que $(\overline{w}, \overline{x}^1, \ldots, \overline{x}^K, \overline{y}^1, \ldots, \overline{y}^K)$ resolve (4.1). Assuma também que $\overline{G}^k = (\overline{V}^k, \overline{E}^k)$ seja o subgrafo de G induzido por $(\overline{x}^k, \overline{y}^k)$, onde $\overline{V}^k = \{i \in V : \overline{y}_i^k > 0\}$ e $\overline{E}^k = \{[i,j] \in E : 0 < \overline{x}_{ij}^k \leq 1\}$. Se, para todo $k \in \mathcal{K}$, $(\overline{x}^k, \overline{y}^k)$ é inteiro e $\overline{G}^k$ não admite subcircuitos, então $(\overline{w}, \overline{x}^1, \ldots, \overline{x}^K, \overline{y}^1, \ldots, \overline{y}^K)$ resolve (3.20). Caso contrário, antes de subdividir o espaço de busca na árvore de enumeração, procuramos identificar desigualdades (3.25) violadas por $(\overline{w}, \overline{x}^1, \ldots, \overline{x}^K, \overline{y}^1, \ldots, \overline{y}^K)$. As desigualdades violadas identificadas são incorporadas a $\mathcal{P}'$ e (4.1) é reotimizado. Este processo segue até que desigualdades *GSEC* violadas não sejam identificadas.

A separação de desigualdades *GSEC* pode ser conduzida em tempo polinomial através da execução de uma série de algoritmos de fluxo máximo (corte mínimo). Mais precisamente, para um dado $k \in \mathcal{K}$, e para cada $i \in V$, procuramos o corte de mínima capacidade que separa 1 de i na rede formada por $\overline{G}^k$ e pelas capacidades $\{\overline{x}_{ij}^k : \ \forall [i,j] \in \overline{E}^k\}$ associadas às suas arestas.

Considere que $[W, \overline{V}^k \setminus W]$ defina o corte de capacidade mínima ($1 \in W$). Sempre que a capacidade $\sum_{[i,j] \in E[W, \overline{V}^k \setminus W]} \overline{x}_{ij}^k$ é menor que $2\overline{y}_z^k$ para $z \notin W$, uma desigualdade *GSEC*

$$\sum_{[i,j] \in E[W, V \setminus W]} x_{ij}^k \geq 2y_z^k, \ \ z \notin W \tag{4.3}$$

é violada por $(\overline{w}, \overline{x}^1, \ldots, \overline{x}^K, \overline{y}^1, \ldots, \overline{y}^K)$.

Em nossa implementação, para cada $k \in \mathcal{K}$, apenas a desigualdade mais violada é inserida em $\mathcal{P}'$. Para a identificação dos cortes de mínima capacidade, empregamos uma implementação do algoritmo *Preflow-Push* descrito em Goldberg e Tarjan [1986].

O algoritmo de separação de *GSECs* é chamado para cada rota $k \in \mathcal{K}$. Assim sendo, a complexidade do problema de separação de *GSECs* é $O(Kn^4)$.

4.2.1 Resultados Computacionais

Nesta Seção, apresentamos os resultados obtidos através do nosso algoritmo BC. Estes resultados são comparados àqueles obtidos pelo algoritmo BB baseado na formulação $\mathcal{P}_{FLUXO}$.

4.2.1.1 Limites de Relaxação Linear

Para efeitos de comparação, obtivemos os resultados da Relaxação Linear (RL) de ambas as formulações para o conjunto de instâncias da TSPLIB utilizados na Seção 4.1. Conforme mencionamos anteriormente, os poliedros $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$ foram acrescidos da restrição (3.31). Cabe mencionar no entanto que a introdução desta restrição nos modelos não altera os valores das correspondentes relaxações.

Na Tabela 4.2, apresentamos os valores das RLs dos dois modelos ($\underline{f}_{FLUXO}$ e $\underline{f}_{GSEC}$ para, respectivamente, $\mathcal{P}_{FLUXO}$ e $\mathcal{P}_{GSEC}$), para $K \in \{1, 2, 3\}$. Apresentamos também os valores dos tempos de CPU (t), em segundos, necessários para avaliá-los. Salientamos que os valores $\sum_{k\in\mathcal{K}} \sum_{i\in V} c_i^k y_i^k$, obtidos quando as relaxações foram determinadas, foram expurgados dos limites apresentados.

Instância	$K=1$				$K=2$				$K=3$			
	$\mathcal{P}_{FLUXO}$		$\mathcal{P}_{GSEC}$		$\mathcal{P}_{FLUXO}$		$\mathcal{P}_{GSEC}$		$\mathcal{P}_{FLUXO}$		$\mathcal{P}_{GSEC}$	
	$\underline{f}_{FLUXO}$	t (s)	$\underline{f}_{GSEC}$	t (s)	$\underline{f}_{FLUXO}$	t (s)	$\underline{f}_{GSEC}$	t (s)	$\underline{f}_{FLUXO}$	t (s)	$\underline{f}_{GSEC}$	t (s)
eil51	25.04	0.66	349.00	0.21	16.09	14.58	180.66	10.06	13.44	18.04	125.63	20.22
eil76	27.08	2.25	438.46	1.23	17.08	25.80	225.35	25.37	14.09	74.75	155.44	55.39
rat99	91.92	10.12	904.75	7.11	54.54	88.98	472.75	46.65	44.18	211.60	331.67	134.11
eil101	27.32	6.21	587.53	2.55	16.69	68.07	299.72	63.28	13.82	251.53	205.34	215.11
bier127	2971.14	19.33	108564.39	21.77	1696.22	136.46	54754.32	291.98	1329.36	359.25	36939.16	894.12
ch130	229.91	20.08	4983.00	17.43	147.70	101.85	2562.04	316.11	120.85	576.85	1755.82	1211.72
pr136	4214.50	25.11	57947.50	17.98	2428.09	331.09	31242.17	362.66	1856.13	783.16	21905.65	1407.57
pr144	3955.88	31.11	45602.75	40.80	2130.99	439.59	23206.18	494.56	1556.86	807.11	15829.76	1962.33
ch150	250.13	45.32	4771.69	37.68	135.70	433.44	2483.35	426.25	98.91	1027.73	1727.49	1896.47
rat195	88.37	129.78	1269.99	76.38	54.23	1066.94	667.15	745.64	42.86	2684.36	464.30	2778.75
tsp225	149.53	185.91	2555.13	227.91	89.33	1618.37	1323.19	1248.42	72.33	11495.35	914.40	12371.34

Tabela 4.2. Limites Inferiores obtidos através da relaxação linear das formulações apresentadas

Como pode ser observado, os limites de RL dados por $\mathcal{P}_{GSEC}$ são sistematicamente mais fortes que os alcançados pela formulação $\mathcal{P}_{FLUXO}$, independente do valor de K. Já os tempos necessários para avaliá-los são de certa forma comparáveis; pelos resultados obtidos é difícil apontar alguma tendência. No caso da formulação $\mathcal{P}_{GSEC}$, em média, apenas 10% do tempo total é gasto separando desigualdades *GSEC*, enquanto o restante é gasto resolvendo os Programas Lineares ao longo do algoritmo de Planos de Corte.

4.2.1.2 Resultados do Algoritmo Branch-and-Cut

Nesta Seção, apresentamos os resultados dos experimentos realizados com a utilização do algoritmo BC. O tempo de execução do BC foi também limitado em no máximo 4 horas. Caso o tempo limite seja alcançado, a execução é interrompida e os limites superiores e inferiores são recuperados. Por fim, a mesma solução inicial fornecida ao algoritmo BB foi também fornecida ao BC.

Os resultados para o algoritmo BC são apresentados na tabela 4.3. A primeira coluna indica a instância correspondente àquela da TSPLIB. Para cada valor de K, são apresentadas quatro colunas. A primeira, intitulada $\overline{f}_{HEU}$, representa a solução fornecida inicialmente para o CPLEX. A segunda coluna, $\overline{f}$, indica o valor da melhor solução encontrada ao final da execução do BC. A terceira coluna, $\underline{f}$, indica os limites inferiores obtidos ao término da execução. Por fim, a quarta coluna apresenta os *gaps* de dualidade quando o algoritmo foi interrompido. Um hífen em uma entrada na coluna *gap* indica que a instância foi resolvida na otimalidade.

Enquanto o algoritmo BB não foi capaz de resolver instância alguma no tempo estabelecido, o algoritmo BC resolveu 3 instâncias para $K = 1$ e uma instância para $K = 2$. O tempo gasto para resolver as 3 instâncias para $K = 1$ foi de aproximadamente 1 minuto. Já a instância *eil51* para $K = 2$ foi resolvida em 12960 segundos.

Além disso, confirmando a tendência apontada na Tabela 4.2, os limites inferiores obtidos pelo BC são sistematicamente mais fortes. Para $K = 1$, o *gap* de dualidade médio ao término (interrupção) da execução para as 11 instâncias foi de 17.88% para o BB e 7.35% para o BC. Para $K = 2$, o *gap* médio foi de 34.89% para ao BB e 16.81% para o BC. Finalmente, para $K = 3$, o *gap* médio obtido foi de 41.44% e 25.25% para, respectivamente, o BB e o BC. Mesmo com esta redução, os *gaps* obtidos ainda são muito altos. Além disso, o BC ainda apresenta dificuldades em encontrar soluções viáveis que aprimorem as soluções iniciais, na medida em que n e K crescem.

4.2.1.3 Comparação entre o Grau de Dificuldade do Algoritmo BC para resolver o PIRC e Problemas Relacionados

Após os resultados apresentados, procuramos avaliar, através do algoritmo BC aqui implementado, o quão difícil na prática é o PIRC quando comparado a outros problemas de Otimização Combinatória relacionados. Para tanto, executamos novas baterias de testes com o objetivo de avaliar duas características essenciais do PIRC: sua função objetivo e sua natureza seletiva.

Com o objetivo de avaliar como a natureza min/max da função objetivo do PIRC dificulta sua resolução exata, investigamos como o algoritmo BC se comporta quando a função objetivo do PIRC é substituída por:

Instância	$K=1$				$K=2$				$K=3$			
	$\overline{f}_{HEU}$	$\overline{f}$	$\underline{f}$	gap (%)	$\overline{f}_{HEU}$	$\overline{f}$	$\underline{f}$	gap (%)	$\overline{f}_{HEU}$	$\overline{f}$	$\underline{f}$	gap (%)
eil51	369	(364)	364.00	-	196	(193)	193.00	-	143	(141)	138.09	2.08
eil76	479	(464)	464.00	-	261	(250)	234.32	6.29	188	188	157.06	16.47
rat99	1067	(1024)	1005.32	1.82	610	(603)	483.26	19.86	507	507	332.79	34.36
eil101	617	(600)	600.00	-	327	(310)	302.19	2.53	238	238	205.50	13.66
bier127	117241	(112066)	111604.61	0.41	60299	(59412)	54876.16	7.63	44082	44082	36946.86	16.19
ch130	5749	(5474)	5259.75	3.91	2999	2999	2567.96	14.37	2240	2240	1757.83	21.53
pr136	76702	(66296)	60434.26	8.84	43520	43520	31396.13	27.86	33209	33209	22548.10	32.10
pr144	57333	57333	46576.00	18.76	34139	34139	23271.64	31.83	28982	28982	17062.46	41.13
ch150	5518	(5322)	4911.83	7.71	3089	3089	2488.33	19.45	2331	2331	1730.89	25.75
rat195	1760	(1729)	1308.29	24.33	993	993	674.58	32.07	778	778	464.72	40.27
tsp225	3170	(3070)	2606.76	15.09	1780	1780	1370.42	23.01	1391	1391	914.47	34.26
Gap médio				7.35				16.81				25.25

Tabela 4.3. Resultados do Algoritmo BC

$$\min \sum_{k\in\mathcal{K}} \sum_{[i,j]\in E} d_{ij} x_{ij}^{k}. \tag{4.4}$$

A Tabela 4.4 apresenta os resultados do algoritmo BC para o novo Problema de Otimização obtido quando o objetivo (4.4) é considerado. O limite de tempo imposto à execução do BC permanece em no máximo 4 horas. Naturalmente, são apresentados resultados para apenas $K \in \{2, 3\}$.

Os resultados obtidos sugerem que a natureza min/max do PIRC dificulta sua resolução exata. Isto parece ser verdadeiro porque o algoritmo BC foi, no mesmo tempo limite de CPU, capaz de apresentar *gaps* de dualidade inferiores para o novo Problema de Otimização. Para $K = 2$, o *gap* médio do BC decresceu de 16.81% para 13.02% com a substituição da função objetivo. Quando o caso $K = 3$ é considerado, a substituição de (3.20) por (4.4) implicou na redução do *gap* médio de 25.25% para 22.13%. Observe também que o tempo de resolução da instância *eil51* para $K = 2$ caiu de 12960 segundos para apenas 301 segundos. Além disso, foi possível resolver esta mesma instância para $K = 3$.

Agora, apresentaremos uma avaliação de outro possível fator complicante do PIRC, sua natureza seletiva. nos experimentos que seguem, foi imposto um raio de comunicação $R = 0$. Desta forma, não há natureza seletiva, pois $\omega(i) = \{i\}\ \forall i \in V$ e, consequentemente, todos os vértices devem ser visitados pelos sorvedouros móveis. Para esta bateria de experimentos, resolvemos o PIRC com sua função objetivo original.

A Tabela 4.5 apresenta os resultados para o PIRC quando $R = 0$. Quando $K = 1$ e $R = 0$ o problema é equivalente ao PCV. Se para os resultados do PIRC com $R > 0$ foi possível resolver apenas 3 instâncias, com $R = 0$ todas as instâncias foram resolvidas

Instância	$K=2$				$K=3$			
	$\overline{f}$	$\underline{f}$	gap (%)	t (s)	$\overline{f}$	$\underline{f}$	gap (%)	t (s)
eil51	375	375.00	-	301.03	390	390.00	-	3632.71
eil76	477	471.76	1.11	-	506	468.70	7.38	-
rat99	1082	963.36	10.97	-	1362	998.80	26.67	-
eil101	613	606.20	1.12	-	705	615.08	12.76	-
bier127	115485	109987.73	4.76	-	132114	110820.48	16.12	-
ch130	5992	5140.65	14.21	-	6644	5268.80	20.70	-
pr136	67114	60508.92	9.84	-	69694	62383.04	10.49	-
pr144	68245	46416.91	31.98	-	86829	47433.34	45.37	-
ch150	5425	4853.06	10.54	-	6958	4937.47	29.04	-
rat195	1978	1323.52	33.09	-	2328	1368.61	41.21	-
tsp225	3558	2644.67	25.67	-	4128	2739.31	33.64	-
Gap médio			13.02				22.13	

Tabela 4.4. Resultados do BC quando a função objetivo (3.20) é substituída por (4.4)

dentro do limite de 4 horas pré-estabelecido. De fato, o tempo máximo necessário para resolver uma instância foi de apenas 472 segundos (*rat195*). Já para $K = 2$, foi possível resolver 5 instâncias e o *gap* médio sofreu uma redução de 16.81% para 1.87%, apenas com a alteração do raio R. Para $K = 3$, não foi possível resolver nenhuma instância, mas o *gap* médio caiu bastante, de 25.25% para 16.82%.

Instância	$K=1$				$K=2$				$K=3$			
	$\overline{f}$	$\underline{f}$	gap (%)	t (s)	$\overline{f}$	$\underline{f}$	gap (%)	t (s)	$\overline{f}$	$\underline{f}$	gap (%)	t (s)
eil51	426	426	-	0.2	223	223.00	-	246	159	153.34	3.58	-
eil76	538	538	-	0.1	277	277.00	-	546	207	186.64	9.85	-
rat99	1211	1211	-	1.0	663	663.00	-	2066	563	440.01	21.85	-
eil101	629	629	-	0.7	326	323.70	0.72	-	242	219.76	9.20	-
bier127	118282	118282	-	3.4	59754	59754.00	-	6981	47621	39994.11	16.02	-
ch130	6110	6110	-	13	3204	3142.03	1.94	-	2453	2113.46	13.84	-
pr136	96772	96772	-	21	53145	51787.38	2.55	-	42534	33272.62	21.77	-
pr144	58537	58537	-	17	34304	34304.00	-	4150	29462	21049.87	28.55	-
ch150	6528	6528	-	19	3391	3332.22	1.74	-	2777	2239.86	19.34	-
rat195	2323	2323	-	472	1251	1186.65	5.15	-	1031	809.86	21.45	-
tsp225	3916	3916	-	154	2199	2012.92	8.46	-	1702	1369.04	19.56	-
Gap médio			-				1.87				16.82	

Tabela 4.5. Resultados do BC para o PIRC com R = 0

Por fim, realizamos experimentos em que tanto R foi fixado em 0 quanto a função objetivo (3.20) foi substituída por (4.4). Através da análise dos dados apresentados na Tabela 4.6, é possível verificar o quanto estes dois fatores em conjunto dificultam a resolução do PIRC. Para $K = 2$, 10 das 11 instâncias foram resolvidas no tempo

limite. Para a única que não foi resolvida, *tsp225*, o *gap* de dualidade no momento da interrupção da execução era de 0.31%. Já para $K = 3$, 9 das 11 instâncias foram resolvidas dentro do tempo limite estabelecido. Dentre as que não foram resolvidas, *rat195* e *tsp225*, os *gaps* de dualidade no momento em que o algoritmo foi interrompido eram, respectivamente, 1.85% e 1.77%.

Instância	$K = 2$				$K = 3$			
	$\overline{f}$	$\underline{f}$	gap (%)	t (s)	$\overline{f}$	$\underline{f}$	gap (%)	t (s)
eil51	438	438.00	-	11	451	451.00	-	30
eil76	548	548.00	-	23	559	559.00	-	51
rat99	1249	1249.00	-	68	1297	1297.00	-	108
eil101	640	640.00	-	94	654	654.00	-	90
bier127	119027	119027.00	-	649	120273	120273.00	-	8654
ch130	6227	6227.00	-	452	6373	6373.00	-	7520
pr136	97488	97488.00	-	236	99176	99176.00	-	769
pr144	59334	59334.00	-	529	60139	60139.00	-	1754
ch150	6560	6560.00	-	519	6609	6609.00	-	1309
rat195	2354	2354.00	-	2647	2430	2385.20	1.85	-
tsp225	3982	3969.62	0.31	-	4115	4042.28	1.77	-
Gap médio			0.03				0.33	

Tabela 4.6. Resultados do BC quando a função objetivo (3.20) é substituída por (4.4) e R = 0

Como pode ser visto, os resultados computacionais que obtivemos sugerem que a conciliação da função objetivo min/max com a natureza seletiva colaboram para uma difícil solução exata para o PIRC. Em particular, o algoritmo BC encontra, em muitos casos, dificuldades para encontrar soluções melhores que aquelas fornecidas pelas heurísticas. Com o intuito de tentar acelerar a obtenção de soluções viáveis de melhor qualidade para o PIRC e, eventualmente, resolver mais instâncias na otimalidade, implementamos um algoritmo Local Branching [Fischetti e Lodi, 2003] que emprega o BC como resolvedor interno. Tal algoritmo é discutido a seguir.

4.3 Um Algoritmo Local Branching

O terceiro algoritmo exato implementado para o PIRC é um algoritmo Local Branching (LB) [Fischetti e Lodi, 2003]. O LB consiste em utilizar um algoritmo para resolver um Problema de Programação Inteira como uma *caixa-preta* para explorar de forma eficaz subespaços de soluções definidos e controlados em um nível estratégico através de um arcabouço externo de ramificação (*branching*). Trata-se da formalização através de modelos de Programação Matemática de procedimentos de Busca Local, comumente presentes em métodos heurísticos de solução de Problemas de Otimização. No LB, as

vizinhanças são exploradas através da imposição de desigualdades lineares ao Programa Inteiro, que restringem o subespaço a ser investigado pelo resolvedor de Problemas de Programação Inteira.

No algoritmo LB aqui implementado, empregamos o algoritmo BC como resolvedor interno. A nossa escolha pelo algoritmo BC baseado na formulação $\mathcal{P}_{GSEC}$, em detrimento do algoritmo BB baseado em $\mathcal{P}_{FLUXO}$, foi motivada pelo fato dos limites de relaxação linear dados por $\mathcal{P}_{GSEC}$ serem mais fortes que os dados por $\mathcal{P}_{FLUXO}$.

Assumindo que $(\tilde{w}, \tilde{x}, \tilde{y})$ denote uma solução inicial viável para o PIRC, e que $\tilde{S}_k := \{i \in V : \tilde{y}_i^k = 1\}, \forall k \in \mathcal{K}$, inicia-se o algoritmo resolvendo-se o subproblema:

$$\min \{w : (w, x, y) \in \mathcal{P}_{GSEC} \cap (4.6) \cap (\mathbb{R}, \mathbb{B}^{K|E|}, \mathbb{B}^{K|V|})\}, \tag{4.5}$$

onde

$$\sum_{k \in \mathcal{K}} \sum_{i \in \tilde{S}_k} (1 - y_i^k) + \sum_{k \in \mathcal{K}} \sum_{i \in V \setminus \tilde{S}_k} y_i^k \leq M \tag{4.6}$$

denota a restrição de Local Branching empregada. Nesta restrição, o parâmetro M determina o tamanho da vizinhança de $(\tilde{w}, \tilde{x}, \tilde{y})$ sendo investigada. Caso a solução $(\hat{w}, \hat{x}, \hat{y})$ de (4.5) seja aprimorante $(\hat{w} < \tilde{w})$, dois novos subproblemas são investigados. Assumindo que $\hat{S}_k := \{i \in V : \hat{y}_i^k = 1\}, \forall k \in \mathcal{K}$, estes subproblemas são:

$$\min \{w : (w, x, y) \in \mathcal{P}_{GSEC} \cap (4.9) \cap (4.10) \cap (\mathbb{R}, \mathbb{B}^{K|E|}, \mathbb{B}^{K|V|})\}, \tag{4.7}$$

e

$$\min \{w : (w, x, y) \in \mathcal{P}_{GSEC} \cap (4.9) \cap (4.11) \cap (\mathbb{R}, \mathbb{B}^{K|E|}, \mathbb{B}^{K|V|})\}, \tag{4.8}$$

onde as restrições de Local Branching

$$\sum_{k \in \mathcal{K}} \sum_{i \in \tilde{S}_k} (1 - y_i^k) + \sum_{k \in \mathcal{K}} \sum_{i \in V \setminus \tilde{S}_k} y_i^k \geq M + 1, \tag{4.9}$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \hat{S}_k} (1 - y_i^k) + \sum_{k \in \mathcal{K}} \sum_{i \in V \setminus \hat{S}_k} y_i^k \leq M \tag{4.10}$$

e

$$\sum_{k \in \mathcal{K}} \sum_{i \in \hat{S}_k} (1 - y_i^k) + \sum_{k \in \mathcal{K}} \sum_{i \in V \setminus \hat{S}_k} y_i^k \geq M + 1 \tag{4.11}$$

são empregadas.

A execução do algoritmo LB então prossegue, resolvendo-se o subproblema (4.7). A exploração dos subespaços não explorados após a resolução de (4.7) segue, de forma análoga. Em nossa implementação, adotamos o valor $M = 5$. Valores maiores para M, de acordo com nossos experimentos, inviabilizam a solução dos subproblemas em tempos aceitáveis. Em nossa implementação, os subproblemas são resolvidos na otimalidade, a não ser que o tempo limite global de 4 horas seja alcançado.

Em nosso estudo, o arcabouço externo de ramificação empregado é aquele implementado por Martinez e Cunha [2009]. Naquele trabalho, um arcabouço genérico de Local Branching foi desenvolvido com o propósito de servir a qualquer Problema de Programação Inteira. Optamos por utilizar este arcabouço em detrimento daquele oferecido pelo CPLEX uma vez que a implementação deste último não permite controlar todos os parâmetros necessários. Por exemplo, não é possível empregar o algoritmo BC para resolver subproblemas no Local Branching oferecido pelo CPLEX. A utilização do BC é necessária, uma vez que necessitamos adicionar cortes *GSEC* para obtermos a formulação $\mathcal{P}_{GSEC}$.

4.3.1 Resultados Obtidos com o Algoritmo Local Branching

Os experimentos realizados com o LB foram conduzidos com configurações semelhantes àqueles realizados para o BC. A solução inicial que foi fornecida como ponto de partida para o BC e o BB foi também utilizada para fazer a primeira ramificação da árvore de enumeração do LB (definir a restrição (4.6)). Além disso, o mesmo tempo limite de execução foi imposto.

A Tabela 4.7, que apresenta os resultados obtidos para o LB, permite uma comparação com as melhores soluções encontradas pelo BC. Para cada valor de $K \in \{1, 2, 3\}$, são apresentadas 4 colunas. Na primeira, apresentamos os limites superiores ($\overline{f}$) encontrados pelo algoritmo BC. As próximas colunas indicam o limite superior, limite inferior ($\underline{f}$) e *gap* de dualidade ao término (ou interrupção) da execução de cada instância para o algoritmo LB.

Os resultados computacionais indicam que o LB conseguiu melhorar a solução inicial mais frequentemente que o BC. Não apenas isto, para $K = 1$, o LB obteve uma solução melhor que o BC em 6 das 8 instâncias para as quais a otimalidade não foi provada, enquanto que o BC obteve uma solução melhor que o LB apenas para a instância *ch150*. O tempo gasto para resolver as 3 instâncias para $K = 1$ foi semelhante ao tempo do BC, aproximadamente 1 minuto. Para $K = 2$, enquanto o BC resolveu a instância *eil51* em 12960 segundos, o LB conseguiu resolvê-la em 5990 segundos. Para as demais instâncias, o LB encontrou soluções melhores que o BC em 7 casos, enquanto que o BC encontrou uma solução melhor que o LB em dois casos. O BC conseguiu melhorar

a solução inicial fornecida pelos procedimentos heurísticos em 5 das 11 instâncias, já o LB encontrou uma solução melhor que a fornecida inicialmente em 10 das 11 instâncias. Finalmente, para $K = 3$, o impacto da utilização do algoritmo LB foi menor, em apenas 2 das 11 instâncias o LB encontrou solução melhor que o BC. Em todas as demais instâncias (excluindo a *eil51*), nem o BC nem o LB conseguiram melhorar a solução inicial fornecida pelos procedimentos heurísticos. Isto ocorre porque o LB encontra grandes dificuldades para, em tempos inferiores ao tempo máximo estabelecido, resolver os subproblemas quando $K = 3$, mesmo para valores pequenos de M, como $M = 5$.

Em contrapartida, os *gaps* de dualidade alcançados pelo BC são melhores, em média, que aqueles alcançados pelo LB para $K = 2$ e $K = 3$. Isto pode ser explicado pela filosofia do LB, que é orientada a buscar soluções aprimorantes mais rapidamente ao explorar subespaços promissores. Devido ao tempo limite imposto de no máximo 4 horas, o tempo restante após a exploração dos subproblemas nem sempre foi suficiente para explorar o último nó da árvore externa do método LB, de modo a alcançar limites duais mais fortes.

	$K=1$				$K=2$				$K=3$			
Instância	BC	LB			BC	LB			BC	LB		
	$\overline{f}$	$\overline{f}$	$\underline{f}$	gap (%)	$\overline{f}$	$\overline{f}$	$\underline{f}$	gap (%)	$\overline{f}$	$\overline{f}$	$\underline{f}$	gap (%)
eil51	364	364	364	-	193	193	193	-	141	141	137.26	2.65
eil76	464	464	464	-	250	249	233.75	6.12	188	185	155.48	15.96
rat99	1024	1024	1005.65	1.79	603	599	476.8	20.4	507	507	332.04	34.51
eil101	600	600	600	-	310	317	300.92	5.07	238	230	205.34	10.72
bier127	112066	11193	111617.92	0.28	59412	59892	54754.3	8.58	44082	44082	35894.52	18.57
ch130	5474	5467	5177.4	5.3	2999	2961	2540.04	14.22	2240	2240	1649.5	26.36
pr136	66296	66116	61803.03	6.52	43520	42061	31041.76	26.2	33209	33209	20741.86	37.54
pr144	57333	56842	48046.75	15.47	34139	34059	22045.26	35.27	28982	28982	12429.08	57.11
ch150	5322	5323	4823.44	9.38	3089	3044	2459.28	19.21	2331	2331	1591.64	31.72
rat195	1729	1706	1287.39	24.54	993	961	660.25	31.3	778	778	442.61	43.11
tsp225	3070	3023	2555.14	15.48	1780	1780	1260.1	29.21	1391	1391	866.3	37.72
Gap médio				7.16				17.78				28.72

Tabela 4.7. Resultados do Algoritmo LB

4.4 Comentários

Os métodos de solução exata apresentados neste capítulo não permitiram, ao nosso ver, resolver de forma satisfatória o PIRC. Mesmo após horas de execução, *gaps* de dualidade ainda elevados foram obtidos para instâncias de dimensões pequenas, quando comparadas às instâncias de dimensões similares resolvidas por outros algoritmos exatos que lidam com problemas de roteamento de um ou mais veículos.

Acreditamos que o desempenho do algoritmo BC (e por consequência do algoritmo LB) poderá ser melhorado através das seguintes ações:

- Implementação de algoritmos de separação para outras classes de desigualdades válidas para o PIRC, que são também válidas para outros problemas de roteamento de veículos, como as desigualdades *Blossom*, *Comb*, etc. [Gendreau et al., 1997].

- Investigação do impacto de otimizarmos sobre o fecho-1 de Chvátal-Gomory associado à formulação $\mathcal{P}_{GSEC}$ [Fischetti e Lodi, 2007; Bonami et al., 2008; Avella et al., 2009].

- Investigação do uso de desigualdades que definem facetas [Cornuéjols e Sassano, 1989; Mannino e Sassano, 1995; Sassano, 1989; Saxena, 2004a,b,c] para o Problema de Recobrimento de Conjuntos em nosso algoritmo BC. Uma vez que observamos que a natureza seletiva do PIRC dificulta sua solução exata, acreditamos que o uso de desigualdades válidas para o Problema de Recobrimento de Conjuntos definido por (3.22) poderá acelerar a resolução do BC.

- Investigação do uso de desigualdades válidas (por exemplo, *extended cover inequalities* [Balas, 1975; Wolsey, 1975]) para politopos da Mochila 0-1 obtidos por meio de uma agregação não negativa (do tipo *surrogate*) das desigualdades (3.22).

Todas estas ações, entretanto, serão conduzidas posteriormente, como continuação desta pesquisa.

Face à nossa dificuldade de resolver o PIRC de forma exata, desenvolvemos heurísticas de solução para o problema. Este é o tema do capítulo que segue.

Capítulo 5

Métodos Heurísticos para o PIRC

Neste capítulo, apresentamos heurísticas para resolver o PIRC. Face aos elevados tempos computacionais requeridos pelos algoritmos exatos anteriormente descritos, o nosso objetivo aqui é propor algoritmos capazes de encontrar soluções viáveis (idealmente de boa qualidade) em tempos de execução baixos o suficiente para permitir o emprego dos métodos propostos em ambientes de simulação de RSSF. Assim sendo, apresentamos algumas heurísticas baseadas em metaheurísticas. Estas heurísticas são compostas por mecanismos de Busca Local / diversificação e um procedimento construtivo, descrito a seguir.

5.1 Heurística Construtiva

A heurística construtiva implementada para o PIRC é baseada no Algoritmo de Inserção do Vizinho mais Próximo (CIA[1]) [Julstrom, 1999] proposta para o PCV Euclideano. Para permitir uma melhor compreensão do algoritmo implementado, vamos primeiramente descrever como o algoritmo CIA opera para o PCV definido em um conjunto de n vértices e matriz de distâncias d. Em seguida, sua adaptação para o PIRC será apresentada.

No caso do PCV, a ideia principal do algoritmo é iterativamente construir uma rota cobrindo os n vértices através de um processo que constrói uma rota com $r \leq n$ vértices a partir de uma rota prévia que possuía $r-1$ vértices. Especificamente, em uma dada iteração do CIA, sejam $S, \overline{S}$ respectivamente o conjunto de nós visitados pela rota na iteração atual e seu complemento em V. Assuma que $p \in V$ é visitado logo após $i \in V$ na rota parcial em construção, isto é, um arco conectando i a p faz parte da rota. A política de seleção que escolhe o vértice a ser inserido na solução parcial é baseada na regra da inserção do menor custo incremental. Para qualquer

[1]Sigla em inglês para *Cheapest Insertion Algorithm*

$j \in \overline{S}$, define-se $\Delta_{ip}^{j} := d_{ij} + d_{jp} - d_{ip}$ como o custo de inserir j entre os vértices i e p. Em conformidade, $\Delta^{j} := \min\{\Delta_{ip}^{j} : i, p \in S,\ p \text{ é visitado logo após } i\}$ denota o mínimo incremento ao comprimento da rota ao se inserir j em S, considerando todas as possíveis posições de inserção. Com base nestas definições, o vértice escolhido para entrar na solução parcial é $z \in \arg\min\{\Delta^{j} : j \in \overline{S}\}$. O algoritmo então insere z entre os vértices i e p para os quais o mínimo Δ^{z} foi atingido e remove z de $\overline{S}$. Este processo continua até que $\overline{S} = \emptyset$.

A adaptação deste procedimento construtivo ao PIRC, produzindo o algoritmo `CIA_PIRC`, é apresentada a seguir. Suponha inicialmente que $K = 1$. Redefiniremos S como o conjunto de *cluster heads* e $\overline{S}$ como o conjunto de vértices que não são cobertos por algum vértice em S, i.e., $\overline{S} = V \setminus \bigcup_{i \in S} \omega(i)$. Para o caso do PIRC, adicionam-se iterativamente novos *cluster heads*, um de cada vez, até que $\overline{S} = \emptyset$.

Este procedimento pode ser facilmente generalizado para valores de $K : K \geq 2$. Neste caso, nosso algoritmo constrói as $K \geq 2$ rotas simultaneamente, adicionando, em cada iteração, um *cluster head* a uma das rotas. Uma vez que o objetivo é construir um conjunto de rotas onde o tamanho da maior delas seja minimizado, sempre inserimos um novo *cluster head* na rota com o menor comprimento na solução parcial. Caso, com a inserção do novo *cluster head*, a rota onde este foi inserido passe a não ser mais a de menor comprimento, procura-se a nova rota mais curta. A política de seleção e o critério de parada são independentes do valor de K.

Porém, na prática, observamos que, para o caso do PIRC, a regra da inserção do menor custo incremental não necessariamente é a melhor política de seleção. Ao se decidir qual vértice será um novo *cluster head* em uma determinada rota, deve-se considerar dois fatores: o custo de expandir a rota e o número de vértices ainda incapazes de se comunicar com o sorvedouro, após a expansão. Nossos resultados computacionais indicaram que a decisão gulosa de escolher o nó que acarreta o menor incremento do comprimento da rota é geralmente dominada (em termos de qualidade da solução) por outra política que potencialmente insira um vértice um pouco mais distante, mas que possua um número maior de vértices ainda não cobertos em seu raio de comunicação. Assim, em nossa implementação do algoritmo `CIA_PIRC`, a política que define qual vértice expandirá uma determinada rota é dada por:

$$z \in \arg\min \left\{ \Delta^{j} - \lambda |\overline{\omega}(j)| : j \in \overline{S} \right\}, \tag{5.1}$$

onde λ é um parâmetro de ajuste no algoritmo e $\overline{\omega}(j) := w(j) \setminus \bigcup_{i \in S} w(i)$ representa o conjunto de nós sensores em $\omega(j)$ que ainda não estão cobertos por algum *cluster head* já presente em alguma das rotas. Empiricamente, concluímos que o parâmetro λ deve depender das propriedades geométricas da área sensoriada. Após a execução de testes

comparativos, definimos que $\lambda = 0.15L$, onde L é o comprimento (em metros) do lado de um quadrado que representa a área sendo monitorada.

O algoritmo `CIA_PIRC`, em conjunto com operadores de Busca Local, permite o desenvolvimento de algoritmos específicos para o PIRC baseados em metaheurísticas. Os operadores desenvolvidos são discutidos na próxima Seção.

5.2 Operadores de Diversificação e Intensificação

Com o intuito de reduzir o comprimento da rota mais longa das soluções iniciais obtidas com o algoritmo `CIA_PIRC`, dois procedimentos de Busca Local (BL) (`2-OPT` [Croes, 1958] e `2-SWAP` [Michiels et al., 2007]) e um mecanismo de diversificação (Algoritmo de Reinserção de Vértices (`ARV`) [de Oliveira et al., 2007]) foram implementados e testados computacionalmente.

5.2.1 2-OPT

Dada uma solução viável para o PIRC, o procedimento `2-OPT` consiste em remover 2 arcos da rota mais longa e, em seguida, tentar reconectar os vértices que estão nas extremidades dos arcos removidos de forma a diminuir o comprimento total da rota. Como, neste trabalho, estamos interessados em minimizar o comprimento da maior rota, o procedimento `2-OPT` é sempre aplicado à maior das K rotas de uma determinada solução. Caso, com a aplicação deste procedimento, a rota seja reduzida o suficiente para que não seja a mais longa, o procedimento passa a operar na nova rota mais longa. A Figura 5.1 ilustra como este procedimento é realizado.

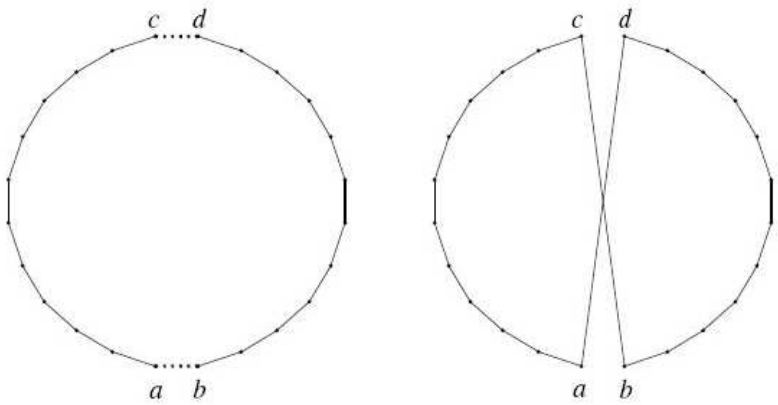

Figura 5.1. Visão geral do procedimento `2-OPT` para uma determinada rota

A vizinhança explorada pelo procedimento `2-OPT` é denominada 2-Exchange. A

avaliação de um movimento na vizinhança 2-Exchange é realizada eficientemente, sem que efetivamente seja feita a reconexão dos arcos. É necessário apenas recuperar os custos das arestas a serem removidas e os custos das arestas a serem inseridas na rota mais longa.

A implementação do procedimento `2-OPT` utiliza a estratégia Primeiro-Aprimorante. Nesta estratégia, assim que uma solução de custo menor que a solução atual é encontrada, o procedimento interrompe a investigação da vizinhança da solução atual e passa a investigar a nova solução de menor custo. Empiricamente, foi verificado que a qualidade final das soluções obtidas com a utilização das duas estratégias mais comuns (Primeiro Aprimorante x Melhor Aprimorante) é semelhante, ao passo que a estratégia Primeiro Aprimorante é mais rápida, uma vez que, na prática, são realizadas poucas varreduras completas na vizinhança da solução sendo investigada.

Além disso, visando reduzir ainda mais os tempos de execução, foi implementada a estratégia de redução de vizinhança *Don't Look Bits* [Bentley, 1990]. Empiricamente, verificamos que a deterioração na qualidade das soluções causada por esta redução de vizinhança, quanto comparada à análise da vizinhança como um todo, é muito pequena e que os tempos de execução são significativamente menores.

Por fim, o procedimento `2-OPT` utiliza indexação circular. Assim que uma solução candidata de custo menor que a atual é encontrada e passa a ser investigada, a varredura da vizinhança inicia-se pelo ponto na rota onde foi realizada a troca de arcos. É válido mencionar que o procedimento mantém um registro do ponto de investigação de cada rota. Assim sendo, durante a aplicação do procedimento, caso uma rota que já havia sido previamente investigada passe a ser novamente a maior rota, o procedimento continuará a investigação da vizinhança 2-Exchange do ponto onde havia parado na investigação anterior.

5.2.2 2-SWAP

Na busca `2-SWAP`, o objetivo é tentar diminuir o custo da solução ao promover a substituição de um *cluster head* pertencente a alguma das rotas por outro *cluster head* (não necessariamente da mesma rota), e vice-versa. Assumindo a existência de dois *cluster heads* $p \in S_{k_1}$ e $q \in S_{k_2}$, deve-se substituir p por q e q por p. Observe que após a permutação dos dois vértices, apenas dois conjuntos de *cluster heads* são alterados: S_{k_1} se transforma em $(S_{k_1} \setminus \{p\}) \cup \{q\}$ e S_{k_2} se transforma em $(S_{k_2} \setminus \{q\}) \cup \{p\}$. Caso $k_1 = k_2$, apenas a ordem de visitação de p e q é trocada. Assim como a implementação de `2-OPT`, este procedimento utiliza a estratégia Primeiro Aprimorante e é implementado com indexação circular.

Cabe destacar que a vizinhança definida pela união das duas vizinhanças men-

cionadas até aqui é *desconexa*, isto é, a partir de uma determinada solução inicial, não é possível alcançar, através da aplicação dos procedimentos de busca que operam sobre tais estruturas, qualquer outra solução do problema. Observe, por exemplo, que qualquer solução vizinha em relação à vizinhança 2-Exchange possui exatamente o mesmo conjunto de *cluster heads* que a solução original. A busca `2-SWAP`, por sua vez, permite alterações no conjunto de *cluster heads*. Porém, não é possível encontrar, em relação a esta vizinhança, soluções vizinhas que possuam um conjunto de *cluster heads* compostos por pelo menos um vértice que não era um *cluster head* na solução original.

Até o momento, não dispomos de uma vizinhança conexa para o PIRC cuja exploração através de um procedimento de Busca Local seja limitado por um polinômio. Desta forma, optamos por implementar um mecanismo de diversificação que permita alterações no conjunto de vértices que determinam o conjunto de *cluster heads* de uma solução viável, oferecendo a possibilidade de exploração de um espaço de busca mais amplo. Este mecanismo é apresentado a seguir.

5.2.3 `ARV` (Algoritmo de Reinserção de Vértices)

O último operador implementado é o Algoritmo de Reinserção de Vértices (ARV) [de Oliveira et al., 2007]. O `ARV` é composto por duas fases. A primeira fase consiste em tentar remover cada *cluster head* de sua rota, de acordo com uma certa probabilidade. Sempre que um *cluster head* $p \in S_k$ (em conjunto com seus arcos incidentes) é removido da rota k, os dois nós vizinhos de p na rota são conectados por um arco, de forma a estabelecer um circuito Hamiltoniano cobrindo o conjunto remanescente $S_k \setminus \{p\}$ de *cluster heads*. O conjunto de rotas obtidas após a possível remoção de cada *cluster head* não é necessariamente viável ($\overline{S}$ pode não ser um conjunto vazio). A segunda fase, então, consiste em aplicar o procedimento `CIA_PIRC` a este conjunto de rotas como medida para recuperar a viabilidade. Ao assim proceder, *cluster heads* diferentes daqueles que foram removidos na fase anterior são possivelmente adicionados e uma nova solução viável é obtida. Assim sendo, o algoritmo `ARV` permite explorar soluções que não poderiam ser alcançadas através da aplicação dos procedimentos `2-OPT` e `2-SWAP`.

Na próxima Seção, mostraremos como são integrados estes três operadores de diversificação / intensificação e o procedimento construtivo `CIA_PIRC`, resultando em diferentes heurísticas para o PIRC.

5.3 Heurísticas Baseadas em Metaheurísticas

Nesta seção, apresentamos como o procedimento construtivo e os operadores descritos previamente são combinados e controlados em arcabouços algorítmicos. Dependendo da forma como os procedimentos descritos nas Seções 5.1 e 5.2 são guiados para explorar o espaço de soluções, obtemos diferentes procedimentos heurísticos: três implementações híbridas da metaheurística GRASP [Feo e Resende, 1995] e uma implementação da metaheurística Busca Local Iterada (ILS[2]) [Martin e Otto, 1996].

5.3.1 GRASP

O GRASP (*Greedy Randomized Adaptive Search Procedure*) é uma classe de algoritmos estocásticos de busca que utilizam heurísticas construtivas gulosas randomizadas para gerar um grande número de soluções candidatas possivelmente distintas para o problema de Otimização que se deseja resolver. Em cada iteração do GRASP, a solução obtida com a fase construtiva randomizada é submetida a um procedimento de busca local [Hoos e Stützle, 2004]. Este processo de duas fases é repetido até que um determinado critério de parada seja satisfeito. Normalmente, este critério é definido por um número máximo de iterações ou um limite de tempo pré-estabelecido.

As três versões híbridas do GRASP propostas neste trabalho fazem uso da mesma implementação randomizada da heurística `CIA_PIRC`, seguida da execução da busca local `2-OPT`. Em cada iteração, assim que um mínimo local em relação à vizinhança 2-Exchange é encontrado, outra metaheurística é aplicada à solução. Assim, dependendo de qual metaheurística é chamada após a execução inicial de `2-OPT`, diferentes versões híbridas do GRASP são obtidas. O algoritmo descrito na Figura 5.2 ilustra os principais passos de nossa abordagem híbrida. Todas as três diferem apenas no passo 7 na Figura 5.2.

Antes de descrever como o ocorre a hibridização do GRASP, vamos primeiro discutir como funciona sua fase construtiva randomizada. Para tanto, considere que $\Delta = \min\left\{\Delta^j - \lambda|\overline{\omega}(j)| : j \in \overline{S}\right\}$ denota o custo mínimo de se adicionar um novo *cluster head* à menor rota em uma dada iteração do procedimento construtivo. Ao invés de adicionar à rota o vértice s para o qual Δ é obtido, na versão randomizada de `CIA_PIRC`, escolhe-se aleatoriamente qualquer nó sensor $j \in \overline{S}$ cujo valor correspondente de $\Delta^j - \lambda|\overline{\omega}(j)|$ pertence ao intervalo $[\Delta, (1+\alpha)\Delta]$, onde o parâmetro $\alpha \geq 0$ controla o nível de randomização do procedimento.

Como em nossa implementação α não permanece constante durante todas as iterações do GRASP; nosso procedimento é, na verdade, um GRASP reativo (veja

[2]Sigla em inglês para *Iterated Local Search*.

```
GRASP/Híbrido
 1  ▷ Parâmetros de entrada: grafo (V, E), matriz de distâncias d
 2  H_melhor ← ∅
 3  f(H_melhor) ← +∞
 4  repeat
 5          Aplique CIA_PIRC randomizado, obtendo a solução H
 6          Aplique Busca Local 2-OPT LS, obtendo o Ótimo Local H'
 7          Aplique outra Metaheurística a H', obtendo H''
 8          if f(H'') < f(H_melhor)
 9            then H_melhor ← H'', f(H_melhor) ← f(H'')
10    until Critério de Parada é satisfeito.
11  return H_melhor, f(H_melhor)
```

Figura 5.2. Descrição do Algoritmo para o GRASP Híbrido

[Prais e Ribeiro, 2000] para detalhes). Em nossa implementação, na primeira iteração do GRASP, α é aleatoriamente escolhido no conjunto $M = \{0.05, 0.10, ..., 0.50\}$ com probabilidades uniformes. À medida que o algoritmo evolui, probabilidades maiores são atribuídas aos valores de α que resultam em soluções de melhor qualidade.

A atualização das probabilidades é realizada da seguinte forma. Seja p_m a probabilidade associada à escolha de α_m, para $m = 1, \ldots, |M|$. Inicialmente, utilizamos os valores iniciais $p_m = 1/|M|, m = 1, \ldots, |M|$. Periodicamente, essa distribuição de probabilidades é atualizada, a partir de informação coletada durante a busca. Considere que $\overline{f}^*$ seja o valor da melhor solução encontrada. Ademais, seja $\overline{f}_m$ o valor médio das soluções encontradas quando $\alpha = \alpha_m$ na fase construtiva. A distribuição de probabilidades é atualizada a cada número pré-determinado de iterações (na nossa implementação, 50). Para fazê-lo, computamos:

$$q_m = \left(\frac{\overline{f}^*}{\overline{f}_m} \right)^{\delta}, \ \forall m \in M.$$

Em seguida, os valores $q_m, m = 1, \ldots, |M|$ são normalizados e a nova distribuição de probabilidades é dada por:

$$p_m = \frac{q_m}{\sum_{j=1}^{j=|M|} q_j}, \ \forall m \in M.$$

Diferentes valores de δ podem ser utilizados para atenuar os valores atualizados das probabilidades p_m. Seguindo recomendações da literatura, neste trabalho escolhemos o valor de $\delta = 10$. Porém, o impacto na qualidade das soluções obtidas para diferentes

escolhas de δ não foi avaliado empiricamente.

5.3.1.1 GRASP-ILS

Em nosso primeiro procedimento GRASP híbrido, denominado GRASP-ILS, uma Busca Local Iterada (ILS) é implementada no passo 7 do algoritmo indicado na Figura 5.2. A principal ideia do ILS é efetuar perturbações em soluções com o objetivo de escapar de mínimos locais. Resumidamente, o ILS é um algoritmo randomizado que primeiramente aplica um procedimento de Busca Local a uma solução viável do problema tratado. Na sequência, o ótimo local em relação à vizinhança da Busca Local aplicada é submetido a um procedimento de perturbação, originando uma solução intermediária. Então, o procedimento de Busca Local é novamente aplicado à solução intermediária até a obtenção de novo mínimo local. Este processo de alternar procedimentos de Busca Local e perturbações é realizado até que um critério de parada seja satisfeito.

No método ILS, a solução que sofrerá perturbação é definida por um critério de aceitação. Esta solução pode ser, por exemplo, a melhor solução já encontrada pelo procedimento. O critério de aceitação é responsável por decidir qual solução passará a ser perturbada nas próximas iterações. Em nossa implementação, a fase ILS do método híbrido sempre perturba a solução inicial fornecida pelo procedimento construtivo após a aplicação de 2-OPT. Mesmo que uma solução melhor que a solução inicial seja encontrada pelo ILS, ainda assim as perturbações serão realizadas sobre a solução inicial. Desta forma, o ILS implementado neste método não possui critério de aceitação, uma vez que sempre a mesma solução é perturbada.

Nesta implementação do GRASP híbrido, a vizinhança 2-SWAP é utilizada na fase de perturbação e o procedimento 2-OPT é utilizado como método de Busca Local interno. Este processo em duas fases é executado para cada solução encontrada pela fase construtiva randomizada, da seguinte forma. Inicia-se a perturbação com apenas uma troca aleatória de vértices (um movimento na vizinhança 2-SWAP). Após um determinado número de iterações sem que haja melhora na solução, aumenta-se o número de perturbações em uma unidade. Toda vez que uma solução melhor é encontrada ou o número de perturbações efetuadas é incrementado, zera-se o contador de iterações sem melhora. Este processo é repetido até que um determinado número máximo de perturbações seja atingido. Assim que são atingidos tanto o número máximo de perturbações permitidas quanto o número máximo de iterações sem melhora, parte-se para a próxima iteração do GRASP.

Ao nosso ver, este procedimento possui como potencial desvantagem o fato de explorar apenas aleatoriamente a vizinhança 2-SWAP. O próximo algoritmo, GRASP-VND, procura contornar esta característica ao promover varreduras sistemáticas desta vizin-

hança.

5.3.1.2 GRASP-VND

GRASP-VND, o segundo procedimento híbrido proposto neste trabalho, implementa, no passo 7 da Figura 5.2, uma adaptação da metaheurística *Variable Neighborhood Descent* (VND) [Mladenović e Hansen, 1997]. Resumidamente, o VND é um método de Busca Local baseado na ideia de explorar, sequencialmente, mais de uma estrutura de vizinhança durante o curso do algoritmo. As estruturas de vizinhança são ordenadas de acordo com sua complexidade: vizinhanças computacionalmente mais baratas de serem avaliadas precedem vizinhanças mais caras.

Nesta metaheurística, primeiro executa-se um algoritmo de Busca Local que explora uma vizinhança mais barata. Assim que um mínimo local com relação a esta vizinhança é encontrado, inicia-se a execução de um novo algoritmo de Busca Local para a próxima vizinhança. Neste processo, assim que uma solução aprimorada é encontrada, a busca recomeça da vizinhança mais barata. O algoritmo termina quando é encontrada uma solução que é mínimo local em relação a todas as vizinhanças consideradas.

Nesta versão do VND, exploramos duas vizinhanças: 2-Exchange e 2-SWAP. Após o passo 6 do algoritmo representado na Figura 5.2, aplicamos 2-SWAP à solução obtida após a aplicação do procedimento 2-OPT. Caso uma solução aprimorada seja encontrada, recorre-se novamente ao procedimento 2-OPT. O VND continua até que a melhor solução em mãos seja um mínimo local de ambas as vizinhanças.

Apesar das duas versões híbridas apresentadas até o momento utilizarem os mesmos dois operadores, GRASP-VND, ao contrário de GRASP-ILS, realiza varreduras sistemáticas nas estruturas de vizinhança consideradas. Por este motivo, tipicamente uma iteração de GRASP-ILS é executada mais rapidamente que uma do GRASP-VND. Consequentemente, para o mesmo tempo de execução, mais soluções iniciais são geradas pelo algoritmo GRASP-ILS, enquanto varreduras mais completas em um número menor de soluções são executadas pela nossa segunda variante híbrida do GRASP.

Porém, como pode ser percebido, nenhum destes métodos propostos utiliza o método ARV. Os próximos métodos, apresentados a seguir, o utilizam.

5.3.1.3 GRASP-ILS/VND

Em nossa terceira e última versão do GRASP, denominada GRASP-ILS/VND, são utilizados, em conjunto com o GRASP, tanto uma versão do ILS quanto uma versão do VND. Nesta implementação, porém, o método ILS, ao contrário do utilizado no algoritmo GRASP-ILS, utiliza o mecanismo de diversificação ARV para perturbar a solução obtida na fase construtiva do GRASP. A implementação do ILS nesta versão também

difere daquela empregada em GRASP-ILS pela escolha do algoritmo de Busca Local. Neste caso, o método VND descrito na Seção anterior substitui a Busca 2-OPT utilizada em GRASP-ILS.

O algoritmo funciona da seguinte forma. Cada solução criada na fase construtiva do GRASP é perturbada por um número fixo de vezes. Após cada perturbação, o VND é aplicado à solução perturbada obtida após a aplicação do ARV. Ou seja, são aplicados à solução perturbada, sequencialmente, os procedimentos 2-OPT e 2-SWAP até que seja obtido um mínimo local relativo às duas vizinhanças correspondentes, 2-Exchange e 2-SWAP, assim como descritos na seção 5.3.1.2. Após a execução do número pré-determinado de perturbações, inicia-se uma nova iteração do GRASP.

5.3.2 ILS

O último método implementado é baseado na metaheurística ILS. Nesta abordagem, uma solução inicial fornecida pela heurística construtiva CIA_PIRC é submetida ao procedimento 2-OPT. Após esta etapa, inicia-se o procedimento ILS com a perturbação sendo realizada por meio da aplicação do mecanismo ARV. Em seguida, novamente aplica-se o procedimento 2-OPT à solução perturbada. Ao contrário do ILS implementado em GRASP-ILS, que sempre perturba a mesma solução, nesta implementação a solução de menor custo encontrada até o momento é a solução que sofre perturbações.

Na próxima Seção, discutiremos os testes realizados para comparar os quatro algoritmos desenvolvidos.

5.4 Resultados Computacionais

Nesta Seção, apresentamos os experimentos computacionais conduzidos para avaliar as heurísticas aqui introduzidas para o PIRC. O propósito é validar um ou mais algoritmos para serem utilizados em um contexto multi-período, como, por exemplo, em um arcabouço de simulação para RSSFs.

Visando avaliar os algoritmos propostos, realizamos testes envolvendo dois grandes grupos de instâncias, cada uma representando uma configuração inicial de uma RSSF. O primeiro grupo de instâncias é o mesmo utilizado para validar os algoritmos exatos propostos no Capítulo 4, composto por instâncias provenientes da biblioteca TSPLIB [TSPLIB, 2009]. A adaptação das instâncias para o PIRC (ao definir R tal que a densidade de comunicação seja o mais próximo possível de 0.75) é realizada assim como foi descrita naquele capítulo.

O segundo conjunto de instâncias aqui considerado é aquele utilizado por Aioffi [2007] para validar o método SHS. Este conjunto é formado por instâncias que pos-

suem n variando de 50 a 600 vértices, aleatoriamente distribuídos sobre uma área de sensoriamento representada por um quadrado no plano. Com o objetivo de testar redes densas e esparsas, mantivemos a área e o raio de comunicação fixos em, respectivamente, $40000m^2$ e $30m$. Para cada valor de n e $K \in \{1, 2, 3, 4\}$, 33 instâncias diferentes foram geradas. Desta forma, os resultados computacionais apresentados para um dado par de n e K são valores médios considerando todas as 33 instâncias.

Para efeitos de comparação, preservamos a melhor solução encontrada pela execução de `CIA_PIRC`. Para tanto, este procedimento foi executado $n-1$ vezes e a melhor solução dentre estas execuções é utilizada na comparação com as heurísticas propostas. As $n-1$ execuções de `CIA_PIRC` diferem na escolha do vértice quanto ao conjunto inicial de *cluster heads* de partida do algoritmo. Em cada execução, a rota inicial é composta pelo depósito e por um nó sensor diferente. Dentre todas estas execuções, a melhor solução obtida é comparada aos valores médios alcançados pelas quatro heurísticas apresentadas na Seção anterior, para os dois grupos de instâncias.

Para cada heurística, foi imposto um tempo limite de 60 segundos. Desta forma, pode-se julgar os méritos de cada algoritmo apenas em termos de qualidade da solução. Ao decidir por este parâmetro, consideramos uma situação realista onde os sorvedouros podem ter um tempo limite de 60 segundos, independente do tamanho da rede, para encontrar a melhor solução possível para o problema, de forma que o tempo empregado na execução do algoritmo não interfira de forma significativa no atraso na entrega de mensagens.

Nas Tabelas 5.1 e 5.2, apresentamos os principais resultados computacionais obtidos por cada heurística para, respectivamente, o primeiro e o segundo grupos de instâncias. Nas primeiras duas colunas da Tabela 5.1, indicamos respectivamente o número de sorvedouros móveis e a instância escolhida da TSPLIB. Na Tabela 5.2, a segunda coluna fornece o tamanho da rede.

Em ambas as tabelas, na terceira coluna (sob o cabeçalho `CIA_PIRC`), apresentamos $\overline{f}$, o comprimento da menor rota mais longa obtida para as $n-1$ execuções independentes de `CIA_PIRC`. Na Tabela 5.2, $\overline{f}$ representa o comprimento médio avaliado para as 33 instâncias do segundo grupo, para cada par n, K. Já na Tabela 5.1, $\overline{f}$ não representa comprimentos médios, uma vez que a execução de `CIA_PIRC` não possui elementos estocásticos e no primeiro grupo cada instância é tratada separadamente.

Nas colunas seguintes, em ambas as Tabelas, apresentamos os resultados médios obtidos por cada heurística. Neste caso, uma vez que as heurísticas apresentam elementos estocásticos, os comprimentos apresentados na Tabela 5.1 são valores médios, avaliados para as 10 execuções de cada instância da TSPLIB para cada valor de K.

Para cada heurística (`GRASP-ILS`, `GRASP-VND`, `GRASP-ILS/VND` e `ILS`), duas entradas

são apresentadas: o comprimento médio $\overline{f}$ da rota mais longa e a redução proporcionada em $\overline{f}$ quando comparada à solução encontrada por `CIA_PIRC`. Por exemplo, na Tabela 5.2, para $K = 1$, $n = 50$, as maiores rotas encontradas por `GRASP-ILS/VND` são, em média, 5.9% mais curtas que as melhores rotas encontradas quando apenas `CIA_PIRC` é utilizada.

Os Resultados na Tabela 5.1 mostram que as menores rotas mais longas, quando comparadas às rotas obtidas pelo uso de `CIA_PIRC`, foram aquelas alcançadas pelo algoritmo `ILS`, quando $K = 1$, e pelo algoritmo `GRASP-ILS/VND`, quando $K \geq 2$. Da mesma forma, os Resultados na Tabela 5.2 indicam que os melhores resultados também foram obtidos pelo algoritmo `ILS`, quando $K = 1$, e pelo algoritmo `GRASP-ILS/VND`, quando $K \geq 2$. Na Tabela 5.2, na medida em que K aumenta de 1 até 4, todas as demais abordagens apresentam resultados melhores que o `ILS`.

Uma diferença importante entre as nossas implementações das metaheurísticas ILS e GRASP é que a ILS opera sempre sobre a melhor solução obtida até aquele instante, enquanto que o GRASP gera novas soluções iniciais a cada iteração. Quando $K = 1$, o `ILS` aparenta alcançar um bom equilíbrio entre intensificação e diversificação nas buscas, principalmente devido ao mecanismo `ARV`. Aparentemente, quando mais sorvedouros são utilizados, o mecanismo `ARV` por si só não foi capaz de prover diversificação suficiente. Nestes casos, a diversificação fornecida pela abordagem *multi-start* do GRASP permitiu a obtenção de melhores rotas. Não obstante, pode-se deduzir que o mecanismo `ARV` possui um impacto considerável nos ganhos obtidos para os métodos `ILS` e `GRASP-ILS/VND` em relação às outras metaheurísticas. Isto parece ser verdade, uma vez que os melhores algoritmos para cada caso ($K = 1$ e $K \geq 2$) implementam `ARV`, ao contrário das outras heurísticas. Este resultado confirma nossas expectativas, uma vez que o mecanismo `ARV` confere diversidade às estruturas de busca empregadas.

Com base nos resultados apresentados, `ILS` e `GRASP-ILS/VND` foram os algoritmos escolhidos (respectivamente para $K = 1$ e $K \geq 2$) para resolver o PIRC em um arcabouço de simulação para RSSFs, discutido no capítulo que segue.

Instância		CIA_PIRC	GRASP-ILS		GRASP-VND		GRASP-ILS/VND		ILS	
K		$\overline{f}$	$\overline{f}$	ganho	$\overline{f}$	ganho	$\overline{f}$	ganho	$\overline{f}$	ganho
1	*eil51*	381.0	366.0	3.9%	366.6	3.8%	366.0	3.9%	366.6	3.8%
	eil76	492.0	480.2	2.4%	484.4	1.5%	480.6	2.3%	480.0	2.4%
	rat99	1114.0	1075.4	3.5%	1081.2	2.9%	1059.4	4.9%	1064.8	4.4%
	eil101	646.0	619.8	4.1%	621.4	3.8%	614.6	4.9%	613.6	5.0%
	bier127	127959.0	115729.4	9.6%	116355.2	9.1%	113856.6	11.0%	113716.0	11.1%
	ch130	6208.0	5624.0	9.4%	5715.4	7.9%	5608.2	9.7%	5643.4	9.1%
	pr136	83730.0	75815.4	9.5%	76947.6	8.1%	74518.0	11.0%	74454.2	11.1%
	pr144	64849.0	57573.2	11.2%	57486.8	11.4%	57044.8	12.0%	57392.0	11.5%
	ch150	6032.0	5587.4	7.4%	5640.8	6.5%	5538.6	8.2%	5554.6	7.9%
	rat195	1854.0	1818.0	1.9%	1823.2	1.7%	1808.8	2.4%	1751.4	5.5%
	tsp225	3277.0	3188.8	2.7%	3199.0	2.4%	3171.6	3.2%	3146.0	4.0%
	Ganho médio			**6.0%**		**5.4%**		**6.7%**		**6.9%**
2	*eil51*	202.0	197.8	2.1%	198.4	1.8%	196.6	2.7%	198.0	2.0%
	eil76	276.0	261.0	5.4%	260.8	5.5%	260.6	5.6%	257.0	6.9%
	rat99	701.0	633.6	9.6%	617.8	11.9%	617.2	12.0%	621.6	11.3%
	eil101	364.0	330.8	9.1%	330.6	9.2%	325.4	10.6%	322.8	11.3%
	bier127	68738.0	61824.4	10.1%	61840.0	10.0%	61184.0	11.0%	60691.2	11.7%
	ch130	3646.0	3239.8	11.1%	3161.4	13.3%	3013.8	17.3%	3152.8	13.5%
	pr136	48194.0	44242.8	8.2%	44573.6	7.5%	43430.8	9.9%	42889.2	11.0%
	pr144	42230.0	37126.6	12.1%	35685.6	15.5%	34341.8	18.7%	36373.8	13.9%
	ch150	3501.0	3194.6	8.8%	3127.4	10.7%	3091.4	11.7%	3105.2	11.3%
	rat195	1032.0	1000.8	3.0%	990.0	4.1%	936.8	9.2%	921.0	10.8%
	tsp225	1895.0	1847.0	2.5%	1863.2	1.7%	1797.2	5.2%	1796.2	5.2%
	Ganho médio			**7.5%**		**8.3%**		**10.3%**		**9.9%**
3	*eil51*	166.0	144.8	12.8%	143.2	13.7%	143.0	13.9%	151.0	9.0%
	eil76	212.0	197.0	7.1%	190.6	10.1%	189.2	10.8%	197.8	6.7%
	rat99	579.0	540.6	6.6%	510.0	11.9%	501.4	13.4%	511.8	11.6%
	eil101	276.0	238.2	13.7%	234.2	15.1%	235.4	14.7%	239.2	13.3%
	bier127	54442.0	46827.0	14.0%	43871.0	19.4%	44334.6	18.6%	44158.8	18.9%
	ch130	2638.0	2428.8	7.9%	2297.4	12.9%	2245.4	14.9%	2325.2	11.9%
	pr136	40024.0	37425.8	6.5%	35859.4	10.4%	33163.0	17.1%	33289.0	16.8%
	pr144	38813.0	31618.2	18.5%	30178.2	22.2%	28981.4	25.3%	31202.6	19.6%
	ch150	2665.0	2425.2	9.0%	2318.4	13.0%	2336.0	12.3%	2388.4	10.4%
	rat195	850.0	826.2	2.8%	801.4	5.7%	777.6	8.5%	782.6	7.9%
	tsp225	1701.0	1583.0	6.9%	1461.8	14.1%	1418.6	16.6%	1450.2	14.7%
	Ganho médio			**9.6%**		**13.5%**		**15.1%**		**12.8%**

Tabela 5.1. Comprimento médio das maiores rotas para as instâncias geradas a partir da TSPLIB

Instância		CIA_PIRC	GRASP-ILS		GRASP-VND		GRASP-ILS/VND		ILS	
K	n	$\overline{f}$	$\overline{f}$	ganho	$\overline{f}$	ganho	$\overline{f}$	ganho	$\overline{f}$	ganho
1	50	793.3	757.3	4.5%	756.0	4.7%	746.2	5.9%	763.0	3.8%
	100	878.9	831.0	5.4%	821.4	6.5%	803.4	8.6%	832.5	5.3%
	150	914.8	869.9	4.9%	859.9	6.0%	837.6	8.4%	848.6	7.2%
	200	914.3	869.8	4.9%	866.4	5.2%	840.6	8.1%	851.6	6.9%
	250	939.8	899.5	4.3%	895.6	4.7%	875.3	6.9%	864.2	8.0%
	300	939.2	903.6	3.8%	901.5	4.0%	885.8	5.7%	869.3	7.4%
	350	953.8	921.2	3.4%	925.2	3.0%	904.3	5.2%	887.6	6.9%
	400	960.6	928.1	3.4%	939.4	2.2%	917.0	4.5%	890.8	7.3%
	450	961.7	935.3	2.7%	945.8	1.7%	923.2	4.0%	887.0	7.8%
	500	968.6	942.2	2.7%	947.8	2.2%	926.0	4.4%	894.7	7.6%
	550	972.1	948.7	2.4%	962.5	1.0%	939.5	3.3%	902.4	7.2%
	600	994.8	963.1	3.2%	977.7	1.7%	954.7	4.0%	912.5	8.3%
		Ganho médio		**3.8%**		**3.6%**		**5.8%**		**7.0%**
2	50	516.2	467.6	9.4%	464.5	10.0%	460.7	10.8%	485.8	5.9%
	100	561.3	512.2	8.7%	506.2	9.8%	495.5	11.7%	520.1	7.3%
	150	573.6	521.2	9.1%	512.8	10.6%	499.6	12.9%	523.1	8.8%
	200	568.3	521.3	8.3%	511.7	10.0%	498.2	12.3%	520.9	8.4%
	250	590.4	541.7	8.3%	532.3	9.8%	513.5	13.0%	536.1	9.2%
	300	589.5	544.7	7.6%	535.6	9.2%	518.7	12.0%	538.4	8.7%
	350	595.9	557.8	6.4%	549.5	7.8%	528.3	11.4%	544.0	8.7%
	400	599.8	562.8	6.2%	554.2	7.6%	537.4	10.4%	547.2	8.8%
	450	598.0	566.3	5.3%	558.8	6.6%	539.4	9.8%	548.0	8.4%
	500	614.2	576.0	6.2%	565.4	7.9%	541.8	11.8%	555.4	9.6%
	550	607.9	578.6	4.8%	568.9	6.4%	545.9	10.2%	554.7	8.7%
	600	619.8	594.8	4.0%	578.2	6.7%	558.1	9.9%	565.9	8.7%
		Ganho médio		**7.0%**		**8.5%**		**11.4%**		**8.4%**
3	50	451.7	397.2	12.1%	395.3	12.5%	393.7	12.8%	419.6	7.1%
	100	480.8	429.4	10.7%	422.9	12.0%	417.1	13.3%	445.5	7.3%
	150	483.1	431.5	10.7%	420.9	12.9%	410.6	15.0%	442.7	8.4%
	200	483.9	433.5	10.4%	422.8	12.6%	410.5	15.2%	446.5	7.7%
	250	505.9	452.6	10.5%	435.1	14.0%	422.3	16.5%	456.2	9.8%
	300	495.5	452.3	8.7%	435.8	12.0%	423.1	14.6%	454.0	8.4%
	350	507.5	467.1	8.0%	449.8	11.4%	433.4	14.6%	465.5	8.3%
	400	512.0	467.6	8.7%	451.4	11.8%	436.6	14.7%	464.9	9.2%
	450	515.2	473.0	8.2%	455.6	11.6%	437.5	15.1%	472.5	8.3%
	500	513.8	474.3	7.7%	452.7	11.9%	435.7	15.2%	468.3	8.8%
	550	513.8	474.7	7.6%	456.2	11.2%	432.0	15.9%	471.5	8.2%
	600	529.5	491.5	7.2%	469.7	11.3%	449.7	15.1%	479.6	9.4%
		Ganho médio		**9.2%**		**12.1%**		**14.8%**		**8.4%**
4	50	412.5	366.8	11.1%	364.8	11.6%	364.4	11.7%	391.4	5.1%
	100	441.7	396.3	10.3%	391.7	11.3%	388.0	12.1%	420.5	4.8%
	150	436.7	390.1	10.7%	382.6	12.4%	376.1	13.9%	406.1	7.0%
	200	438.5	394.3	10.1%	386.0	12.0%	377.1	14.0%	411.5	6.2%
	250	461.5	411.4	10.9%	395.6	14.3%	386.5	16.2%	420.0	9.0%
	300	450.5	407.6	9.5%	393.9	12.6%	383.4	14.9%	415.4	7.8%
	350	468.9	420.0	10.4%	405.2	13.6%	392.6	16.3%	435.3	7.2%
	400	466.6	423.3	9.3%	406.0	13.0%	391.5	16.1%	431.7	7.5%
	450	467.7	426.5	8.8%	407.6	12.8%	392.7	16.0%	429.8	8.1%
	500	473.7	425.1	10.3%	407.3	14.0%	389.7	17.7%	434.5	8.3%
	550	457.3	420.4	8.1%	395.3	13.6%	379.3	17.1%	421.8	7.8%
	600	478.0	438.9	8.2%	419.6	12.2%	402.9	15.7%	439.8	8.0%
		Ganho médio		**9.8%**		**12.8%**		**15.1%**		**7.2%**

Tabela 5.2. Comprimento médio das maiores rotas para as instâncias de Aioffi [2007]

Capítulo 6

Simulação de uma RSSF

Em muitos sistemas dinâmicos complexos, como em uma RSSF, é muito difícil capturar exatamente as interações e relações entre as entidades do sistema através de equações matemáticas. Sendo assim, optamos por desenvolver um Simulador de Eventos Discretos para avaliar como o modelo proposto para a organização da rede se compara com outras abordagens da literatura. Ao longo do capítulo, descrevemos o simulador implementado e como os algoritmos de otimização apresentados anteriormente nos permitiram melhorar parâmetros de QiS em RSSFs.

6.1 Aspectos Gerais do Simulador

Um Simulador de Eventos Discretos foi a ferramenta escolhida para avaliar como a aplicação do PIRC em RSSFs se compara a outras abordagens na literatura, em termos de atraso na entrega de mensagens, tempo de vida da rede e cobertura. Para tanto, desenvolvemos um arcabouço para RSSFs implementado sobre o simulador JIST/SWAMS [JIST, 2007]. Os protocolos de energia, transmissão, sensoriamento e armazenamento são previamente disponíveis na biblioteca de funções do simulador empregado. Além disso, na expectativa de melhorar o comportamento geral da rede, incorporamos outras funcionalidades importantes ao arcabouço: algoritmos para resolver tanto o PIRC quanto o Problema de Controle de Densidade (PCD). Neste modelo de simulação, o PCD é resolvido através da utilização do algoritmo proposto por Aioffi [2007], da forma como foi explicado na Seção 2.2.1.

O modelo de simulação de RSSFs que apresentamos compreende uma série de operações cíclicas realizadas por cada sorvedouro. O fluxograma de atividades realizadas em cada um destes ciclos é apresentado na Figura 6.1. O primeiro ciclo de simulação inicia com a resolução do PIRC, logo após a distribuição dos nós sensores pela rede. Assume-se que os sorvedouros conhecem a posição geográfica de todos os nós sensores.

Após a definição do primeiro conjunto de rotas, o PCD é resolvido. Neste momento, é feita a escolha do conjunto de nós que será mantido ativo durante aquele ciclo, mas a ordem de ativação / desativação ainda não é implementada.

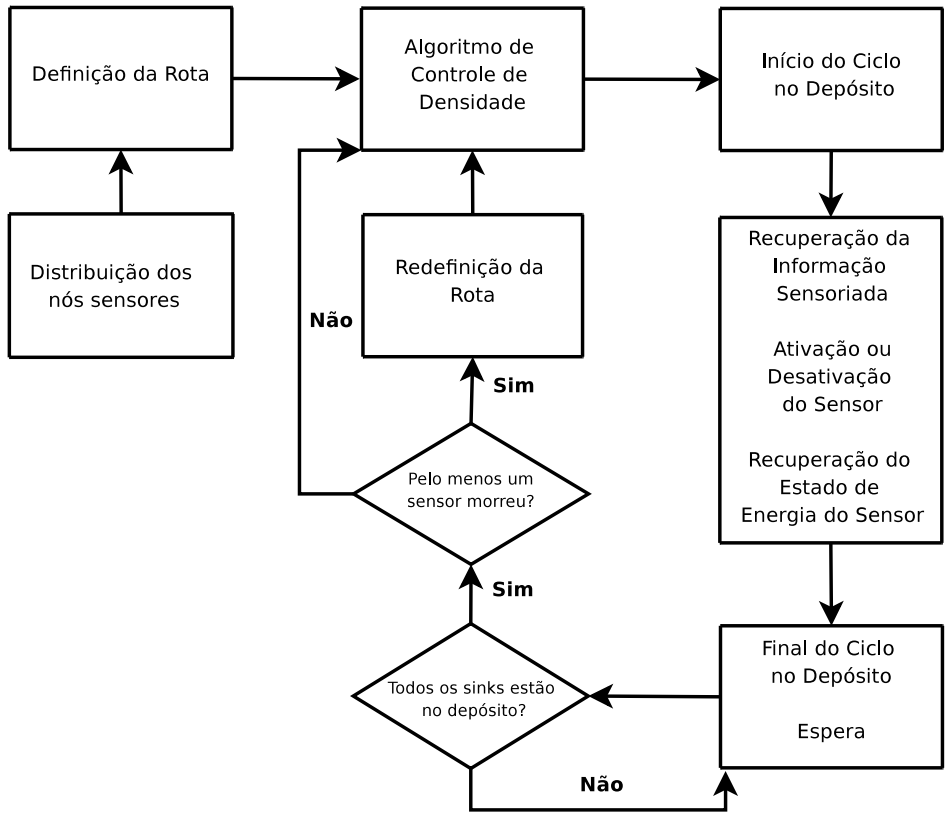

Figura 6.1. Fluxograma do Simulador

Logo após a resolução do PCD, cada sorvedouro inicia sua trajetória pela rede. Por utilizar tecnologias recentes e protocolos suportados para a camada MAC [Polastre et al., 2004; Correia et al., 2005], nós sensores podem ser mantidos em níveis de potência muito baixos (estado inativo). Mesmo em estado inativo, os nós sensores são capazes de retomar seu funcionamento pleno assim que um estímulo externo apropriado é recebido. Por consequência, durante o movimento pela rede, os sorvedouros não apenas coletam a informação sensoriada como também ativam e desativam alguns nós sensores, de acordo com a política de controle de densidade. Neste processo, os sorvedouros também tomam conhecimento do estado de energia de todos os nós sensores nesta rota, informação que será útil posteriormente, quando o novo PCD for resolvido.

No momento em que todos os sorvedouros retornam ao depósito, uma razoável aproximação do estado de energia de todos os nós é disponível. Caso tenha-se verificado

a morte de algum nó sensor (sem energia disponível) durante o ciclo anterior, resolve-se novamente o PIRC, desta vez tendo como conjunto de vértices de entrada apenas os sensores que não estiverem mortos. A nova determinação de rotas possui como objetivo diminuir a rota mais longa, uma vez que os nós sensores mortos não mais se comunicam com os sorvedouros. Entretanto, o novo conjunto de rotas é apenas efetivamente implementado caso a rota mais longa obtida seja menor que a rota mais longa calculada anteriormente.

De posse do estado de energia da rede quando todos os sorvedouros estão no depósito, independente do PIRC ter sido resolvido ou não, o PCD é novamente resolvido. A partir deste momento um novo ciclo de simulação é iniciado. Uma vez que as rotas são determinadas antes da execução do algoritmo de controle de densidade, uma desvantagem da nossa abordagem é que os nós sensores que permaneçam inativos por dois ou mais ciclos consecutivos continuam a ser visitados pelos sorvedouros. Poderia-se justificar esta visita aparentemente desnecessária, por exemplo, pela existência de outros nós, cuja visita é necessária, no *cluster* onde um vértice inativo por dois ou mais ciclos se encontra. Entretanto, se as rotas fossem calculadas novamente sem incluir o vértice inativo, dando origem a novos *clusters*, possivelmente a rota poderia ser mais curta. Se o PCD fosse executado primeiramente, o PIRC poderia excluir da rota aqueles nós sensores que já estavam inativos e que permanecerão inativos no próximo ciclo. Em compensação, esta decisão de implementação (PIRC antes do PCD) permite que tenhamos um melhor conhecimento do estado de energia da rede. Consideramos que melhores políticas de controle de densidade podem ser implementadas se uma informação mais realista do estado de energia estivesse disponível. Esta é a razão pela qual resolvemos o PIRC antes do PCD.

A resolução do PCD também se beneficia do fato dos sorvedouros móveis serem sincronizados. Ainda que este fato possa resultar em maiores taxas de atraso na entrega de mensagens, devido ao fato de que sorvedouros permanecem no depósito à espera da chegada do último sorvedouro, a sincronização permite uma fácil implementação das políticas centralizadas de controle de densidade. A abordagem centralizada potencialmente beneficia o comportamento geral da RSSF. Em nosso modelo, um dos K sorvedouros resolve tanto o PIRC quanto o PCD e, em seguida, informa aos demais tanto as suas novas rotas quanto quais vértices serão visitados e quais serão ativados / desativados durante o próximo ciclo.

Outro fator que pode aumentar a taxa de atraso na entrega de mensagens é o tempo gasto para rodar os algoritmos de otimização, uma vez que o relógio da simulação não para enquanto tais procedimentos são executados. Assim, um limite superior de tempo é imposto ao algoritmo que resolve o PIRC. No primeiro ciclo, é permitido que

algoritmo gaste não mais que 60 segundos. Sempre que houver necessidade de executá-lo novamente (se algum nó sensor morrer), um limite de tempo menor, de apenas 2 segundos, é imposto.

Pode-se imaginar que os 60 segundos necessários para calcular a rota no primeiro ciclo afetam muito negativamente o atraso na entrega de mensagens. Entretanto, uma vez que este cálculo só acontece uma vez, seu impacto no primeiro ciclo (que não deve ser negligenciado) será diluído pelos próximos ciclos. Como nossos resultados computacionais demonstram, o ganho obtido em termos de atraso na entrega de mensagens ao utilizar nossos algoritmos de otimização mais que compensa este tempo inicial.

Os parâmetros de simulação escolhidos neste trabalho foram definidos de acordo com as características dos nós sensores Mica2 [XBOW, 2006], comercialmente disponíveis (ver Tabela 6.1). Para a aplicação particular aqui tratada, foi assumido que os nós sensores periodicamente coletam informação de temperatura codificadas em 32 bits (4 bytes), a uma taxa constante de 1/20Hz (três coletas por minuto). Uma vez que suas placas de memória armazenam até 4 Kbytes (4096 bytes) de informação, conclui-se que os nós sensores possuem autonomia de até aproximadamente 5:40 horas de dados coletados. A camada MAC é a IEEE 802.11, disponível no simulador SWANS, uma vez que os nós sensores Mica2 implementam o protocolo CSMA/CA.

Parâmetro	Valor	Potências	Valor
Energia inicial do sensor	50 mA-hr	Potência gasta na transmissão	8.9 mA
Raio de sensoriamento	15 m	Potência gasta na recepção	7 mA
Área sensoriada	$40000m^2$	Potência gasta com rádio ativo	7 μA
Velocidade do sorvedouro	1 m/s	Potência gasta em processamento	8 mA
Raio de comunicação	$30m$	Potência gasta em sensoriamento	5 mA
Largura de Banda	250 kbps		

Tabela 6.1. Principais parâmetros de simulação

6.2 Resultados Computacionais

Com base nos resultados apresentados na Seção 5.4, `ILS` e `GRASP-ILS/VND` foram os algoritmos escolhidos para resolver o PIRC no arcabouço de simulação, respectivamente, quando $K = 1$ e $K \geq 2$.

Na sequência, avaliamos como uma RSSF organizada de acordo com o modelo PIRC se compara à abordagem SHS, proposta por Aioffi [2007]. As comparações são baseadas em três importantes métricas de RSSFs: a taxa de atraso na entrega de mensagens, a taxa de cobertura e o tempo de vida da rede. Para avaliar as taxas de atraso na entrega de mensagens e o tempo de vida da rede, o tempo máximo de simulação foi definido como 15 horas. Para avaliar a cobertura da rede, este parâmetro foi estendido para 25

horas. Tempos de simulação mais longos foram utilizados neste caso porque concluímos que, após 15 horas, a cobertura ainda era alta para todos os métodos, independente da abordagem utilizada (SHS ou PIRC).

6.2.1 Atraso na Entrega de Mensagens

A taxa de atraso é um parâmetro crucial em várias aplicações importantes, tais como sistemas de controle de incêndios, por exemplo. Nestes casos as taxas de atraso devem ser baixas, uma vez que pode não haver o tempo necessário para a realização de ações de combate quando um foco de incêndio é detectado. Em aplicações onde a informação sensoriada é gerada regularmente, ou a altas frequências (ao invés de ser gerada sob demanda ou baseada em eventos), taxas de atraso na entrega de mensagens devem também ser baixas, evitando a perda de dados devido à baixa capacidade de memória dos nós sensores.

A Figura 6.2 ilustra como o atraso na entrega de mensagens se comporta em função do número de nós sensores. Cinco curvas são apresentadas na Figura 6.2: uma para a abordagem SHS e uma curva para cada valor de K utilizado no modelo PIRC. Ao se comparar os resultados do SHS com aqueles obtidos pelo método PIRC quando $K = 1$, pode-se observar que o último alcançou resultados significativamente melhores. Para $n = 600$, por exemplo, as taxas médias de atraso na entrega de mensagens obtidas pelo PIRC são 23% menores que as taxas obtidas pelo método SHS. Como pode-se verificar na Figura, a taxa de crescimento do atraso na entrega de mensagens na medida em que n aumenta também é muito menor para o PIRC. Observe que quando $n = 50$, ambas abordagens apresentam taxas de atraso similares. Em compensação, na medida em que n aumenta até 600, a taxa média de atraso alcançada pelo PIRC cresce apenas 51%, praticamente metade dos 94% de aumento observados quando o SHS é utilizado.

A utilização de mais sorvedouros móveis, como previsto, também ajuda na redução das taxas médias de atraso. Reduções significativas foram obtidas quando um único sorvedouro móvel foi substituído por mais sorvedouros. Para $n = 600$, por exemplo, as taxas médias de atraso na entrega de mensagens para $K = 2, 3$ e 4 são respectivamente 51%, 63% e 68% menores quando comparados ao caso em que $K = 1$. Pode-se observar também que a taxa de crescimento do atraso decresce na medida em que K aumenta. Quando n aumenta de 50 a 600, a taxa de atraso cresce, para $K = 2, 3$ e 4, respectivamente, 27%, 19% e 17%. Em relação à taxas de atraso, quanto maior o valor de K, mais escalável é a rede à medida que n aumenta.

É importante salientar que ambas as abordagens comparadas aqui, SHS e PIRC, utilizam um protocolo de comunicação *single-hop* baseado em TDMA (*Time Division Multiplexing Access*) e coordenado pelos sorvedouros móveis. Assim, colisões de men-

sagens não podem ocorrer e mensagens só podem ser perdidas quando um nó sensor tem sua bateria descarregada ou quando seus *buffers* estão completamente ocupados. Uma vez que a taxa de atraso máxima na entrega de mensagens entre todos os métodos testados é muito menor que o limite de 5:40 horas de autonomia de sensoriamento, nenhuma mensagem é perdida devido a falta de capacidade de armazenamento.

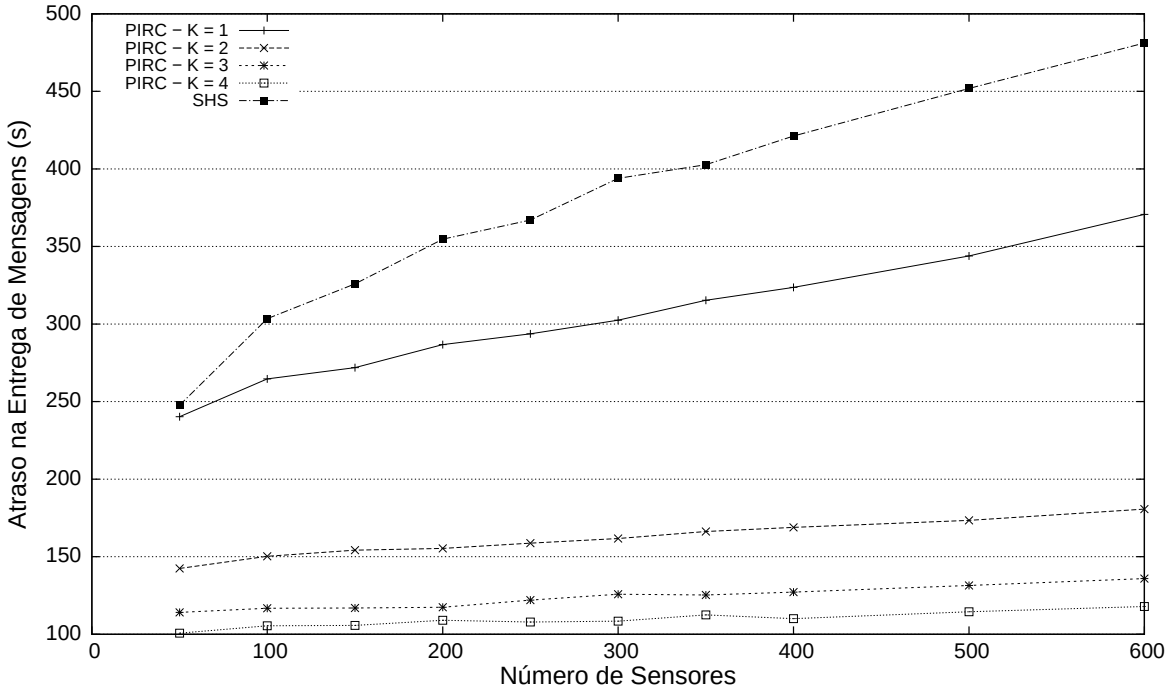

Figura 6.2. Atraso Médio na Entrega de Mensagens, SHS x PIRC/K

Agora, vamos considerar o impacto da utilização de melhores ou piores algoritmos de otimização (com relação ao comprimento da rota mais longa) na taxa de atraso. Com este objetivo, simulamos o comportamento da rede quando `CIA_PIRC` (sob limites de tempo de CPU semelhantes) substitui `ILS` (para $K = 1$) e `GRASP-ILS/VND` (para $K \geq 2$) como método de resolução do PIRC no simulador. Observamos que, quando $K = 1$, a simples substituição do melhor algoritmo de otimização (`ILS`) pelo procedimento `CIA_PIRC` no arcabouço de simulação resultou em taxas médias 12% maiores. De forma análoga, a substituição do algoritmo `GRASP-ILS/VND` por `CIA_PIRC` resultou em taxas médias de atraso 19%, 26% e 27% mais altas para, respectivamente, $K = 2, 3$ e 4. Estes resultados sugerem que o desenvolvimento de algoritmos de otimização de boa qualidade para a resolução do PIRC compensa pelos ganhos obtidos em termos de QoS.

6.2.2 Cobertura da Rede

O próximo parâmetro de QoS avaliado neste trabalho é a taxa de cobertura da rede, i.e., a porcentagem da área monitorada que é sensoriada por pelo menos um nó sensor em um determinado instante de tempo de simulação. A Figura 6.3 indica como a cobertura da rede se altera à medida que o relógio da simulação prossegue (por no máximo 25 horas), para redes com $n = 400$.

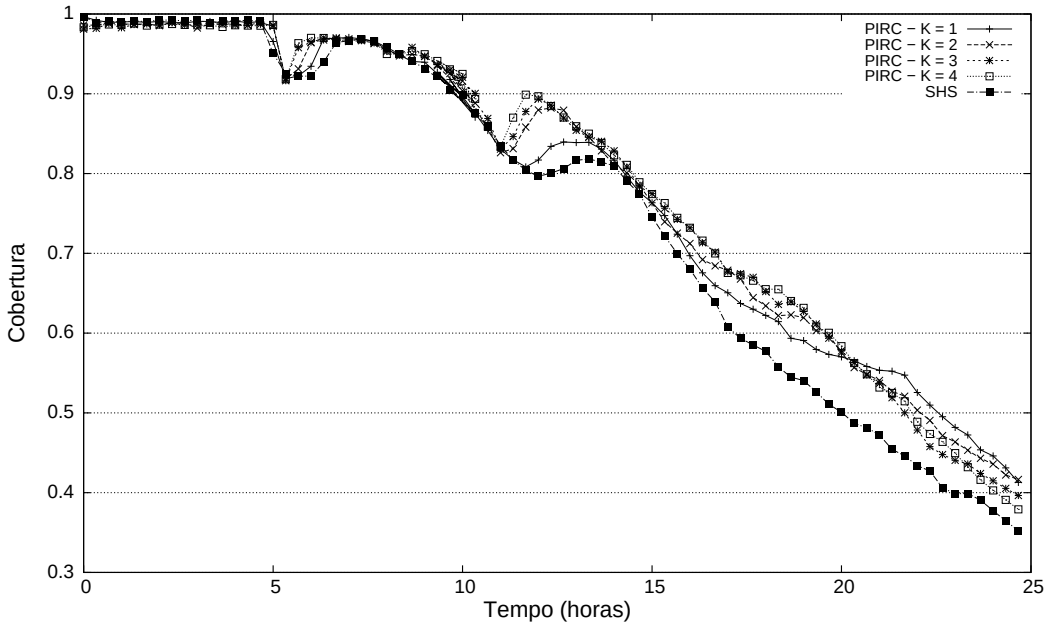

Figura 6.3. Cobertura da Rede (n = 400)

A utilização do PIRC com $K = 1$ permitiu a obtenção de um aumento de 17% em cobertura quando comparado ao SHS (42% de cobertura contra 36%), ao final do tempo de simulação. A razão para este fenômeno está relacionada à frequência com que as políticas de controle de densidades são implementadas. Uma vez que as rotas mais longas do PIRC são muito menores que as equivalentes do SHS, a quantidade de ciclos de simulação realizados em um mesmo período de tempo tende a ser maior para o PIRC. A execução de mais ciclos durante o mesmo intervalo de tempo permite que o algoritmo de controle de densidade seja executado mais frequentemente, sendo assim mais eficiente.

De forma análoga, o tempo necessário entre a decisão do algoritmo de controle de densidade e a execução da ordem de ativação / desativação dos nós sensores pelos sorvedouros móveis tende a diminuir. A decisão do algoritmo é baseada em um certo estado de energia, e no momento em que as ordens são implementadas, este estado de energia sofreu alterações. Quanto mais rápida é a implementação das ordens, menor é

a discrepância do estado de energia da rede.

Outro aspecto que é válido mencionar é que, após cerca de 5 horas de simulação, as taxas de cobertura tanto para o PIRC quanto para o SHS caem de praticamente 100% para algo próximo de 90% e, em seguida, pouco tempo depois, sobem novamente. Estas quedas nas taxas de cobertura ocorrem quando o primeiro nó sensor morre. Assim, as taxas de cobertura permanecem um pouco mais baixas até que todos os sorvedouros retornem à base, o PCD é executado novamente e novas políticas de controle de densidade são então implementadas. Uma vez que as rotas do PIRC são menores, a rede se mantém sob taxas menores de cobertura por períodos mais curtos de tempo. Isto é confirmado pela Figura, uma vez que todas as curvas sofrem uma queda mais ou menos ao mesmo tempo, mas o PIRC com $K = 4$ recupera a cobertura mais rapidamente que os outros métodos. Quedas similares na cobertura, seguidas de recuperações rápidas, ocorrem novamente por volta de 11 horas de simulação.

Contrariamente ao que se pode esperar, a utilização de mais sorvedouros móveis não melhora as taxas de cobertura sistematicamente, para períodos mais longos de simulação. Explicações para este fato serão fornecidas na próxima Seção, em conjunto com a análise do tempo de vida da rede.

6.2.3 Tempo de Vida da Rede

Na literatura, métricas diferentes tem sido utilizadas para medir o quão eficiente é a rede em termos de consumo de energia. De acordo com alguns autores [Gandham et al., 2003], a métrica mais adequada para medir a eficiência energética é dependente da aplicação. Algumas métricas de avaliação consideradas normalmente na literatura são: o tempo de vida da rede (definido como o tempo decorrido até que o primeiro nó sensor morra) e a energia total consumida durante um período pré-determinado de tempo.

Neste trabalho, avaliaremos estas duas métricas. Como será mostrado, elas devem ser avaliadas em conjunto para melhor compreender as vantagens e desvantagens, caso existam, do método aqui proposto. Na Figura 6.4, apresentamos como o tempo de vida da rede varia em função de n e, na Figura 6.5, retratamos o percentual da energia total da rede disponível durante o curso da simulação. A Figura 6.5 foi gerada para $n = 400$.

Como pode ser observado na Figura 6.4, o tempo de vida da rede mantém-se praticamente inalterado quando o número de nós sensores varia entre 50 e 400. Após este ponto, um aumento acentuado no tempo de vida pode ser verificado. Considerando o tamanho da área escolhida para nossas instâncias, redes com 400 ou menos nós são ainda um pouco esparsas. Assim, é bastante provável que um certo ponto de demanda (representado por um ponto discretizado da área monitorada) seja coberto por apenas

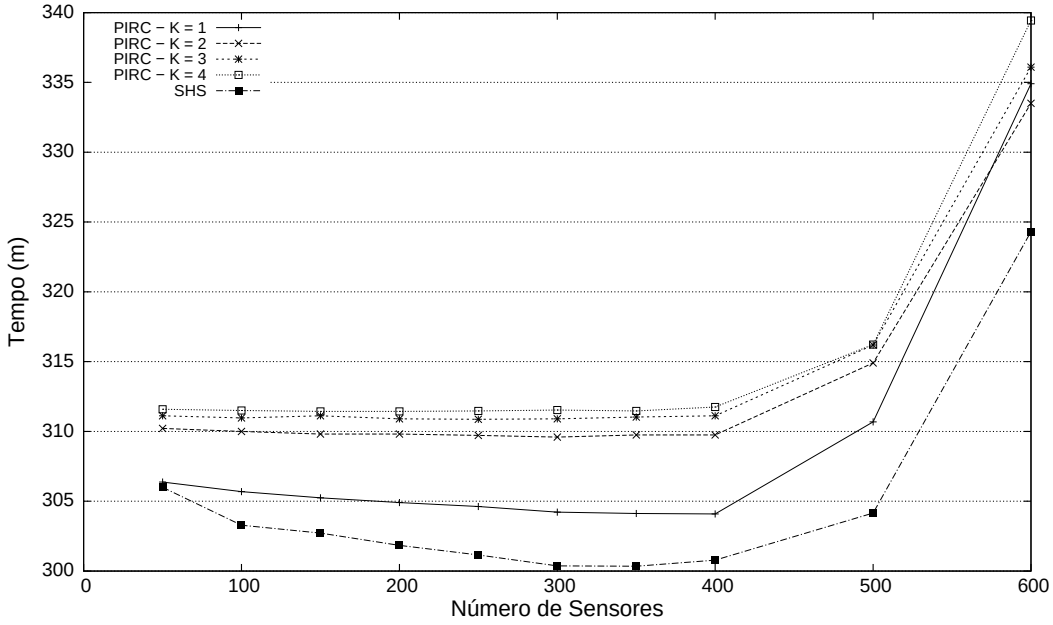

Figura 6.4. Tempo de Vida da Rede

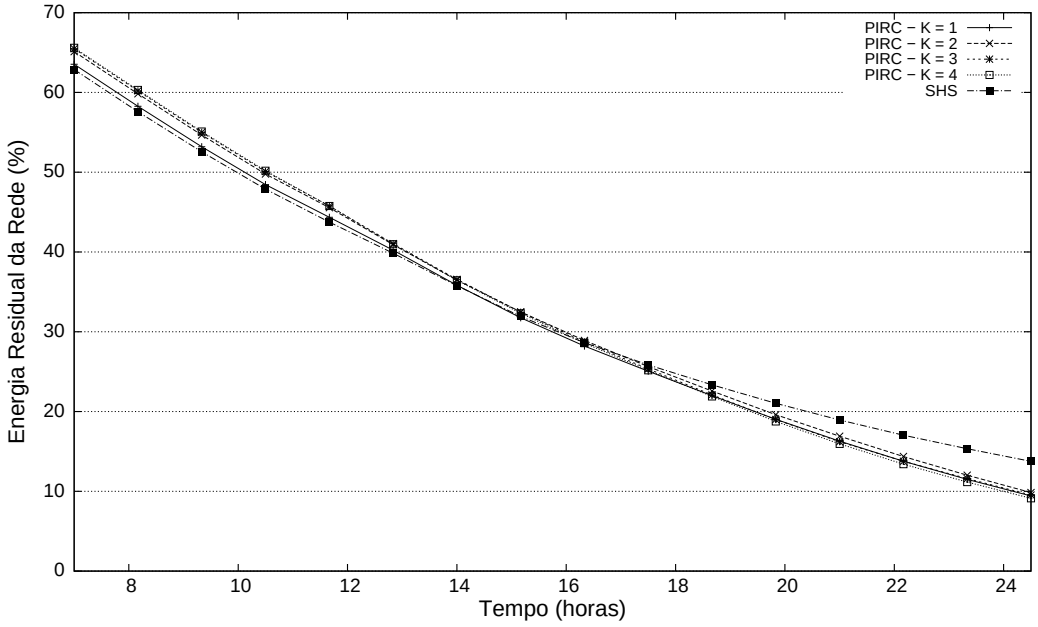

Figura 6.5. Energia Residual da Rede (n = 400)

um nó sensor. Em tese, este nó permanece ativo durante todo o tempo de simulação, uma vez que não há outro nó que cubra aquele mesmo ponto de demanda. Desta forma, os tempos de vida médios para redes entre 50 e 400 nós sensores é semelhante, pois em todas há normalmente a presença do nó sensor que permanece ativo em todo o tempo e morre mais rapidamente que os outros nós. Em compensação, para val-

ores maiores de n, praticamente todo ponto de demanda é coberto por mais de um nó sensor. Consequentemente, a política de controle de densidade implementada neste trabalho permite um revezamento de nós que devem permanecer ativos para todos os pontos de demanda, aumentando assim o tempo de vida da rede.

Observe que o PIRC alcança um tempo de vida da rede maior que o SHS, para todos os valores de n. Assim, com base apenas no tempo de vida da rede, poderia-se dizer que o PIRC alcançou maiores reduções em consumo de energia quando comparado ao SHS. Para melhor compreender esta questão, dois outros parâmetros devem ser considerados: a fração da energia total inicial disponível durante o curso da simulação (veja a Figura 6.5) e o percentual de nós sensores mortos em função do tempo (veja a Figura 6.6). As Figuras 6.5 e 6.6 indicam que o consumo de energia do PIRC é maior quando comparado ao SHS. O número de nós sensores mortos também é maior quando o PIRC é utilizado.

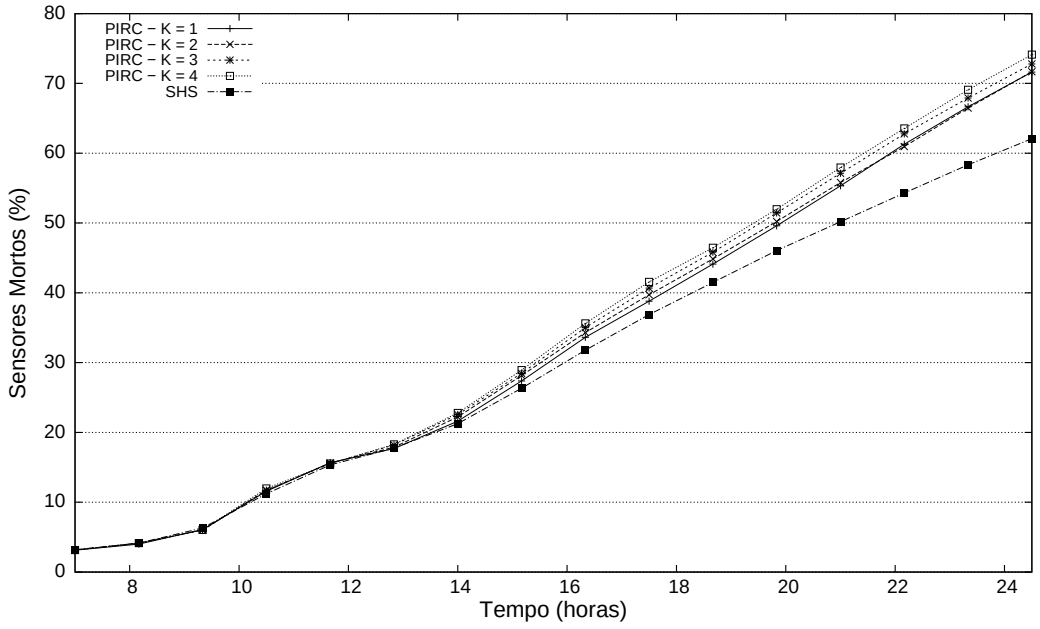

Figura 6.6. Porcentagem de nós sensores mortos (n = 400)

Um consumo mais alto de energia ocorre uma vez que as rotas do PIRC implicam na comunicação mais frequente entre os nós sensores e os sorvedouros. Uma vez que, para a aplicação aqui considerada, a informação é sensoriada a taxas baixas (32 bits a 1/20 Hz), apenas uma pequena fração da memória do nó sensor é efetivamente utilizada em cada transmissão. Porém, há um tamanho mínimo do pacote utilizado na transmissão de dados. Em ambos os métodos (SHS ou PIRC), devido à baixa taxa de sensoriamento, este tamanho mínimo do pacote nunca é suplantado pelo volume de dados sensoriados. Logo, o pacote transmitido nos dois métodos possui sempre o mesmo tamanho. Uma

vez que as rotas do PIRC são menores que as do SHS, no PIRC ocorrem comunicações nó sensor / sorvedouro com mais frequência e, consequentemente, há um maior consumo de energia, pois os pacotes possuem o mesmo tamanho. Este fato também explica a razão das taxas de cobertura decrescerem quando mais sorvedouros são utilizados no PIRC. Claramente, estes experimentos mostram como apenas o tempo de vida da rede não é suficiente para capturar com precisão o comportamento da rede em termos de eficiência energética.

O ganho do método SHS em termos de consumo de energia não se traduz em maior taxa de cobertura da rede, como era de se esperar. A execução mais frequente das políticas de controle de densidade apresentam um impacto maior sobre a taxa de cobertura. Ou seja, há menos nós sensores disponíveis, porém estes são melhor distribuídos pela rede de forma a garantir maiores taxas de cobertura.

Devemos apontar ainda que a vantagem do SHS sobre o PIRC em termos de menor consumo de energia tende a desaparecer em outras aplicações, onde mensagens são geradas a taxas maiores e, consequentemente, a memória dos nós sensores se torna uma restrição mais ativa. Além disso, para a aplicação deste estudo, o PIRC poderia ser facilmente adaptado para alcançar um melhor equilíbrio entre consumo de energia e taxas de atraso. Isto poderia ser obtido impondo um tempo mínimo de espera no depósito para cada sorvedouro móvel, antes do início de um novo ciclo de simulação.

Finalmente, devemos salientar que, quando as taxas de atraso na entrega de mensagens são analisadas em conjunto com parâmetros de energia, o compromisso entre consumo de energia e atraso na entrega de mensagens se torna bastante claro. Para redes com $n = 400$, as taxas médias de atraso para o PIRC com $K = 4$ são da ordem de apenas 25% das taxas correspondentes para o SHS, enquanto que 50% da energia remanescente do SHS estava disponível para o PIRC ao final do tempo de simulação. Considerando que nosso principal objetivo era reduzir as taxas de atraso na entrega de mensagens sem acarretar perdas excessivas de cobertura, afirmamos que o PIRC permitiu alcançar os objetivos aos quais nos propusemos.

Capítulo 7

Conclusão e Trabalhos Futuros

Neste trabalho, procuramos aprimorar parâmetros de Qualidade de Serviço (QoS) em Redes de Sensores Sem Fio (RSSFs) com sorvedouros móveis ao introduzir novos modelos e algoritmos de otimização que tratam de forma integrada a organização destas redes. O principal problema aqui discutido é aquele de definir rotas para os sorvedouros móveis de forma a garantir a comunicação entre os sorvedouros e todos os nós sensores da rede. Este problema é modelado como uma variante do Problema de Roteamento de Veículos não capacitado. Embora muitos trabalhos na literatura procurem estabelecer estratégias de organização para RSSFs com sorvedouros móveis, aparentemente esta variação ainda não havia sido estudada.

O modelo que aqui propomos consiste em encontrar rotas, uma para cada sorvedouro móvel, de forma que todos os nós sensores sejam visitados por um sorvedouro ou estejam *suficientemente próximos* de algum nó sensor visitado por um sorvedouro. Assim, é possível estabelecer a comunicação direta entre os sorvedouros e os nós sensores. Uma observação importante é que, para favorecer o balanceamento da rede, buscamos com que todos os sorvedouros possuam rotas com aproximadamente o mesmo comprimento. Desta forma, o objetivo do problema aqui tratado consiste em minimizar a maior das rotas. Este problema, denominado Problema Integrado de Roteamento e Clusterização (PIRC), foi formulado como um Problema de Otimização em Grafos. Além da formulação em Grafos, duas formulações de Programação Inteira foram apresentadas. A primeira é uma formulação de Fluxos e a segunda é uma formulação baseada em Desigualdades de Eliminação de Subrotas (*GSEC*).

As formulações de Programação Inteira apresentadas foram avaliadas empiricamente através da implementação de algoritmos exatos específicos para cada uma. A formulação de Fluxos foi avaliada através de um algoritmo Branch-and-Bound. Já a formulação baseada em desigualdades *GSEC*, é avaliada através de um algoritmo Branch-and-Cut, onde desigualdades *GSEC* são separadas como planos de corte. Além

do Branch-and-Cut, um algoritmo do tipo Local Branching que o emprega como resolvedor interno foi desenvolvido e avaliado.

Uma importante possibilidade de investigação futura consiste em aprimorar o algoritmo Branch-and-Cut aqui implementado. Isto poderá ocorrer através do uso de desigualdades válidas para o problema também válidas e já caracterizadas para problemas correlatos, do uso de planos de cortes gerais (cortes de Chvátal-Gomory) e do uso de desigualdades oriundas de um estudo poliedral que pretendemos realizar para o PIRC.

Em vista da dificuldade em resolver o PIRC através dos algoritmos exatos mencionados, várias heurísticas baseadas em Metaheurísticas foram propostas para obter soluções viáveis (idealmente de boa qualidade) em tempos de execução aceitáveis, para permitirem seu uso nos processos de decisão em RSSFs. Estas heurísticas são baseadas em metaheurísticas como o GRASP, o ILS e o VND e são compostas por um procedimento construtivo, mecanismos de Busca Local e diversificação.

Um simulador de eventos discretos foi implementado para avaliar como uma RSSF se comporta ao incorporar as heurísticas apresentadas para o PIRC. As simulações realizadas mostraram que ao modelar a definição das rotas dos sorvedouros como o PIRC e resolver este problema através das heurísticas aqui apresentadas, foi possível aprimorar vários parâmetros de QoS em RSSFs quando comparado à outros trabalhos da literatura. A utilização de nossos algoritmos possibilitou consideráveis melhoras em termos de atraso na entrega de mensagens. Além disso, obtivemos ganhos em termos de tempo de vida útil da rede e taxas de cobertura.

Nas simulações realizadas, entretanto, não ocorrem perda de dados por falta de capacidade de memória dos nós sensores, uma vez que as taxas de sensoriamento utilizadas foram relativamente baixas. Pretendemos avaliar o impacto do PIRC em simulações com cenários onde este fato pode ocorrer. Além disso, adicionaremos fatores complicantes ao ambiente de simulação, como obstáculos no terreno da rede e alterações não-programadas nas posições dos nós sensores após o início da simulação, e avaliaremos formas de adaptar o PIRC a redes com estas dificuldades adicionais.

Por fim salientamos que, como fruto de resultados obtidos neste trabalho, foram publicados e submetidos os trabalhos: [Arbex et al., 2008b,a, 2009a,b].

Referências Bibliográficas

Ahuja, R.; Magnanti, T. L. e Orlin, J. B. (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, New Jersey.

Aioffi, W. M. (2007). Métodos Integrados para Organização de Redes de Sensores sem Fio com Sink móvel e Controle de Densidade. Master's thesis, UFMG.

Akyildiz, I. F.; Su, W.; Sankarasubramaniam, Y. e Cayirci, E. (2002). Wireless Sensor Networks: A Survey. *Computer Networks*, 38:393–422.

Al-Karaki, J. N. e Kamal, A. (2004). Routing Techniques in Wireless Sensor Networks: a Survey. *IEEE Wireless Communications*, 11(6):6–28.

Arbex, C.; Cunha, A. S. e Aioffi, W. M. (2008a). Optimization Algorithms for Improving the Quality of Service in Wireless Sensor Networks with Mobile Sinks. In *XL SBPO Simpósio Brasileiro de Pesquisa Operacional*, João Pessoa, PB, Brasil.

Arbex, C.; Cunha, A. S.; Aioffi, W. M. e Mateus, G. R. (2008b). Algorithms for Improving the Quality of Service in Wireless Sensor Networks with Multiple Mobile Sinks. In *MSWIM '08: Proceedings of the 11th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pp. 239–243, Vancouver, BC, Canada. ACM Press.

Arbex, C.; Cunha, A. S.; Mateus, G. R. e Aioffi, W. M. (2009a). Optimization and Simulation in Wireless Sensor Networks with Multiple Mobile Sinks. *Networks*. Submetido para publicação.

Arbex, C.; Cunha, A. S.; Mateus, G. R. e Martinez, L. C. (2009b). Exact Algorithms for a Selective Vehicle Routing Problem where the Longest Route is Minimized. In *LAGOS 2009 - V Latin-American Algorithms, Graphs and Optimization Symposium*. Submetido para publicação.

Avella, P.; Boccia, M. e Vasiliev, I. (2009). Computational Experience with General Cutting Planes for the Set Covering Problem. *Operations Research Letters*, 37:16–20.

Balas, E. (1975). Facets of the Knapsack Polytope. *Mathematical Programming*, 8(1):146–164.

Bentley, J. L. (1990). Experiments on Traveling Salesman Heuristics. In *SODA '90: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 91–99, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Bonami, P.; Cornuéjols, G.; Dash, S.; Fischetti, M. e Lodi, A. (2008). Projected Chvátal-Gomory Cuts for Mixed Integer Linear Programs. *Mathematical Programming*, 113(2):241–257.

Chakrabarti, A.; Sabharwal, A. e Aazhang, B. (2003). Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks. In *The 2nd Intl. Workshop on Information Processing in Sensor Networks (IPSN)*.

Cornuéjols, G. e Sassano, A. (1989). On the 0, 1 Facets of the Set Covering Polytope. *Mathematical Programming*, 43(1):45–55.

Correia, L. H. A.; Macedo, D. F.; Silva, D. A. C.; dos Santos, A. L.; Loureiro, A. A. F. e Nogueira, J. M. S. (2005). Transmission Power Control in MAC Protocols for Wireless Sensor Networks. In *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 282–289, New York, NY, USA. ACM Press.

Croes, G. A. (1958). A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6(6):791–812.

Dantzig, G.; Fulkerson, D. e Johnson, S. (1954). Solution of a Large-scale Traveling Salesman Problem. *Operations Research*, 2:393–410.

Dantzig, G. e Hamser, R. (1959). The Truck Dispatching Problem. *Management Science*, 6:80–91.

de Oliveira, H. C. B.; Vasconcelos, G. C.; Alvarenga, G. B.; Mesquita, R. V. e Souza, M. M. (2007). A Robust Method for the VRPTW with Multi-Start Simulated Annealing and Statistical Analysis. In *SCIS '07: IEEE Symposium on Computational Intelligence in Scheduling*, pp. 198–205.

Feo, T. A. e Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133.

Figueiredo, C.; Nakamura, E. e Loureiro, A. (2004). Protocolo Adaptativo Híbrido para a Disseminação de Dados em Redes de Sensores Sem Fio Auto-Organizáveis. *22o Simpósio Brasileiro de Redes de Computadores*, 1:43–56.

Fischetti, M.; Gonzalez, J. e Toth, P. (1997). A Branch-and-cut Algorithm for the Symmetric Generalized Traveling Salesman Problem. *Operations Research*, 45(3):378–394.

Fischetti, M. e Lodi, A. (2003). Local Branching. *Mathematical Programming B*, 98:23–47.

Fischetti, M. e Lodi, A. (2007). Optimizing over the First Chvátal Closure. *Mathematical Programming*, 110(1):3–20.

Gandham, S.; Dawande, M.; Prakash, R. e Venkatesan, S. (2003). Energy Efficient Schemes for Wireless Sensor Networks With Multiple Mobile Base Stations. In *IEEE GLOBECOM*, San Fransisco, USA.

Gendreau, M.; Laporte, G. e Semet, F. (1997). The Covering Tour Problem. *Operations Research*, 45(4):568–576.

Gendreau, M.; Laporte, G. e Semet, F. (1998). A Branch-and-cut Algorithm for the Undirected Selective Traveling Salesman Problem. *Networks*, 32:263–273.

Glaab, H. (2002). A New Variant of a Vehicle Routing Problem: Lower and Upper Bounds. *European Journal of Operational Research*, 139:557–577.

Goldberg, A. V. e Tarjan, R. E. (1986). A New Approach to the Maximum Flow Problem. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 136–146, New York, NY, USA. ACM.

Grötschel, M.; Jünger, M. e Reinelt, G. (1984). A Cutting Plane Algorithm for the Linear Ordering Problem. *Operations Research*, 32(6):1195–1220.

Heinzelman, W.; Chandrakasan, A. e Balakrishnan, H. (2002). An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Trans. on Wireless Communications*, 1(4):660–670.

Hoos, H. H. e Stützle, T. (2004). *Stochastic Local Search - Foundations and Applications*. Elsevier.

ILOG Cplex Solver (2009). http://www.ilog.com/products/cplex/.

J. E. Kelley, J. (1960). The Cutting-Plane Method for Solving Convex Programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712.

Jain, S.; Shah, R.; Brunette, W.; Borriello, G. e Roy, S. (2006). Exploiting Mobility for Energy Efficient Data Collection in Wireless Sensor Networks. *Mobile Networks and Applications*, 11(3):327–339.

Jea, D.; Somasundara, A. e Srivastava, M. (2005). Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks. In *IEEE/ACM International Conference on Distributed Computing in Sensor Systems*.

JIST (2007). Java in Simulation Time - Scalable Wireless Ad Hoc Network Simulator. http://jist.ece.cornell.edu/.

Julstrom, B. A. (1999). Coding TSP Tours as Permutations Via an Insertion Heuristic. In *SAC '99: Proceedings of the 1999 ACM Symposium on Applied Computing*, pp. 297–301, San Antonio, Texas, United States. ACM Press.

Jünger, M.; Reinelt, G. e Rinaldi, G. (1995). The Traveling Salesman Problem. In M.O. Ball, editor, *Handbooks in OR and MS, Vol. 7*, pp. 225–330. Elsevier Science Publishers.

Khepera-III (2009). http://www.k-team.com.

Kim, H. S.; Abdelzaher, T. F. e Kwon, W. H. (2003). Minimum-energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 193–204, New York, NY, USA. ACM.

Land, A. H. e Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3):497–520.

Luo, J. e Hubaux, J.-P. (2005). Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks. In *The 24th IEEE INFOCOM*.

Machado, M. V.; Goussevskaia, O.; Mini, R. A. F.; Rezende, C. G.; Loureiro, A. A. F.; Mateus, G. R. e Nogueira, J. M. S. (2005). Data Dissemination in Autonomic Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 23(12):2305–2319.

Magnanti, T. e Wolsey, L. (1995). *Optimal trees, In: Network Models, Handbook in Operations Research and Management Science*, pp. 503–615. North-Holland.

Mannino, C. e Sassano, A. (1995). Solving Hard Set Covering Problems. *Operations Research Letters*, 18:1–5.

Martin, O. C. e Otto, S. W. (1996). Combining Simulated Annealing with Local Search Heuristics. *Annals of Operations Research*, 63:57–75.

Martinez, L. C. e Cunha, A. S. (2009). Um Arcabouço Local Branching para Problemas de Otimização Combinatória aplicado ao Problema da Árvore de Custo Mínimo com k arestas. *XLI Simpósio Brasileiro de Pesquisa Operacional*. Submetido para publicação.

Michiels, W.; Aarts, E. e Korst, J. (2007). *Theoretical Aspects of Local Search (Monographs in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Mirchandani, P. e Francis, R. (1990). *Discrete Location Theory*. John Wiley and Sons.

Mladenović, N. e Hansen, P. (1997). Variable Neighborhood Search. *Comps. in Opns. Res.*, 24:1097–1100.

Nakamura, F.; ao, F. Q.; Menezes, G. e Mateus, G. (2005). An Optimal Node Scheduling for Flat Wireless Sensor Networks. In *Lecture Notes in Computer Science, Vol. 3420*, pp. 475–482.

Padberg, M. e Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems. *SIAM Rev.*, 33(1):60–100.

Perkins, C. E. (2001). *Ad Hoc Networking*. Addison-Wesley Professional.

Polastre, J.; Hill, J. e Culler, D. (2004). Versatile Low Power Media Access for Wireless Sensor Networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 95–107, New York, NY, USA. ACM Press.

Prais, M. e Ribeiro, C. C. (2000). Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment. *INFORMS Journal on Computing*, 12:164–176.

Romer, K. e Mattern, F. (2004). The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications*, 11(6):54–61.

Sassano, A. (1989). On the Facial Structure of the Set Covering Polytope Dimensional Linear Programming. *Mathematical Programming*, 44(2):181–202.

Saxena, A. (2004a). On the Set Covering Polytope: I. All the Facets with Coefficients in 0, 1, 2, 3. GSIA Working Paper 2004-E37.

Saxena, A. (2004b). On the Set Covering Polytope: II. Lifting Facets with Coefficients in 0, 1, 2, 3. GSIA Working Paper 2004-E30.

Saxena, A. (2004c). On the Set Covering Polytope: III. Complete Characterization of 3-Critical Matrices. GSIA Working Paper 2004-E31.

Shah, R. C.; Roy, S.; Jain, S. e Brunette, W. (2003). Data MULEs: Modeling a Three-Tier Architecture for Sparse Sensor Networks. pp. 30–41. IEEE.

Siqueira, I.; Fiqueiredo, M.; Loureiro, A.; Nogueira, J. M. S. e Ruiz, L. (2006). An Integrated Approach for Density Control and Routing in Wireless Sensor Networks. In *Parallel and Distributed Processing Symposium*, pp. 10–19, Greece.

Slijepcevic, S. e Potkonjak, M. (2001). Power Efficient Organization of Wireless Sensor Networks. In *ICC 2001: IEEE International Conference on Communications*, volume 2, pp. 472–476, Finland.

Somasundara, A. A.; Ramamoorthy, A. e Srivastava, M. B. (2007). Mobile Element Scheduling with Dynamic Deadlines. *IEEE Transactions on Mobile Computing*, 6(4):395–410.

TSPLIB (2009). http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

Wang, W.; Srinivasan, V. e Chua, K. (2005a). Using Mobile Relays to Prolong the Lifetime of Wireless Sensor Networks. In *MobiCom '05*, pp. 270–283, New York, NY, USA. ACM Press.

Wang, Z. M.; Basagni, S.; Melachrinoudis, E. e Petrioli, C. (2005b). Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime. *Hawaii International Conference on System Sciences*, 9:287a.

Wolsey, L. (1975). Faces for a Linear Inequality in 0-1 Variables. *Mathematical Programming*, 8(1):165–178.

XBOW (2006). MICA2 - Wireless Measurement System. Source: http://www.xbow.com/.

Zhang, H. e Hou, J. (2005). Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. *Ad Hoc & Sensor Wireless Networks*, 1(1-2).