OBERLAN CHRISTO ROMÃO

MÉTODOS PARA REDUÇÃO INTEGRADA DO CONSUMO DE ENERGIA E DO ATRASO NA ENTREGA DE DADOS EM REDES DE SENSORES SEM FIO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2013

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T

R761m
2013

Romão, Oberlan Christo, 1986-

Métodos para redução integrada do consumo de energia e do atraso na entrega de dados em redes de sensores sem fio / Oberlan Christo Romão. – Viçosa, MG, 2013.

xi, 72 f. : il. (algumas color.) ; 29 cm.

Orientador: André Gustavo dos Santos.

Dissertação (mestrado) - Universidade Federal de Viçosa.

Referências bibliográficas: f. 68-72.

1. Cluster (Sistema de computador). 2. Programação linear. 3. Algoritmo genético. 4. Otimização Combinatória. 5. GRASP (Sistema operacional de computador). I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22 ed. 004.35

OBERLAN CHRISTO ROMÃO

MÉTODOS PARA REDUÇÃO INTEGRADA DO CONSUMO DE ENERGIA E DO ATRASO NA ENTREGA DE DADOS EM REDES DE SENSORES SEM FIO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 26 de julho de 2013.

Luciana Brugiolo Gonçalves

Thiago Ferreira de Noronha

Stenio Sã Rosário Furtado Soares

André Gustavo dos Santos
(Orientador)

Dedico este trabalho aos meus pais, irmãos, namorada e amigos que de muitas formas me incentivaram e ajudaram para que fosse possível a concretização deste trabalho.

Agradecimentos

Em primeiro lugar, agradeço a Deus por todas as alegrias, pela saúde e pela força e determinação que me concedeu, para que conseguisse chegar até aqui. Agradeço por sempre colocar pessoas tão especiais a meu lado.

Agradeço também aos meus pais e irmãos por sempre acreditarem em mim e pelo apoio em todos os momentos. Sem vocês, nada disso seria possível. Aos meus pais, obrigado por toda a educação que recebi e pela oportunidade que me deram de realizar meus sonhos. Vocês são exemplos de vida para mim.

A minha namorada e grande amiga Fabíola, por sempre acreditar em mim e me incentivar. Pelo companheirismo, pela paciência, pela amizade, amor e carinho. Sem ela este trabalho e minha vida não seriam completos.

Ao meu orientador André Gustavo dos Santos que colaborou de forma fundamental nesse trabalho, acreditando sempre nas minhas ideias, indicando sugestões que contribuíram de forma significativa. Obrigado pelos ensinamentos, atenção, amizade e dedicação ao longo deste trabalho.

As meus amigos, agradeço por todos os momentos divertidos, de alegria e descontração. Obrigado por sempre torcerem pelo meu sucesso.

A todos os meus professores que são os maiores responsáveis por eu estar concluindo esta etapa da minha vida, compartilhando a cada dia os seus conhecimentos.

Obrigada a todos vocês por participarem desta minha etapa, pois direta, ou indiretamente me fizeram crescer, tanto pessoalmente como profissionalmente.

"May the Force be with us"

Sumário

Lista de Figuras **vi**

Lista de Tabelas **viii**

Lista de Abreviaturas **ix**

Resumo **x**

Abstract **xi**

1 Introdução **1**
1.1 Objetivos . . . 3
1.2 Principais contribuições . . . 4
1.3 Organização da dissertação . . . 4

2 Rede de Sensores Sem Fio **6**
2.1 Conceitos e Aplicações . . . 6
2.2 Trabalhos da literatura em RSSF . . . 9
2.2.1 Disseminação da Informação . . . 9
2.2.2 Controle de Densidade . . . 11
2.2.3 Mobilidade do Sorvedouro . . . 12
2.3 Contribuições . . . 14
2.4 Comentários Finais . . . 16

3 PARST **17**
3.1 O Modelo Proposto . . . 17
3.2 Modelo de consumo de Energia . . . 18
3.3 Uma Formulação Matemática para o PARST . . . 21
3.4 Aumentando o tempo de vida da RSSF . . . 26
3.4.1 Problema Mestre . . . 27
3.4.2 Subproblema . . . 28
3.4.3 Branch-and-Price . . . 28
3.5 Comentários Finais . . . 29

4 Método híbrido para o PARST **30**
4.1 Herística para a floresta de comunicação . . . 31
4.1.1 Método *H-passos* . . . 31
4.1.2 Método *Prim-H* . . . 32
4.2 Heurística para a rota do agente móvel . . . 34
4.3 Herísticas para a seleção dos *cluster heads* . . . 35

4.3.1 Algoritmo Genético 35
4.3.2 GRASP 39
4.4 Busca Local 40
4.4.1 Busca Local para os *Cluster Heads* 41
4.4.2 Busca Local para a rede de comunicação 41
4.4.3 Busca Local para a rota do agente móvel 42
4.5 Pseudo-código final das Metaheurísticas 43
4.5.1 Algoritmo Genético 43
4.5.2 GRASP 43
4.6 Comentários finais 45

5 Resultados Computacionais **46**
5.1 Método exato 49
5.2 Algoritmo Genético 50
5.3 GRASP 51
5.4 Algoritmo Genético vs. GRASP 54
5.5 Maximizando o tempo de vida da RSSF 56

6 Considerações Finais **65**
6.1 Conclusões 65
6.2 Trabalhos futuros 66
6.3 Publicações 67

Referências Bibliográficas **68**

Lista de Figuras

1.1 Rede de Sensor Sem Fio 2

2.1 Modelo de nó sensor Mica2 6
2.2 Exemplo de aplicação de uma RSSF para o controle militar 7
2.3 Topologia de uma RSSF baseada em agrupamentos 10
2.4 Mensagem percorrendo a RSSF em esquema *multi-hop* 10
2.5 Avião do projeto WiFly 12
2.6 Exemplo da topologia da rede SHS 13
2.7 Exemplo da topologia da rede MHS-3 14
2.8 Exemplo da topologia adotada com restrição de saltos H = 3 15
2.9 Agente Móvel recebendo os dados de um cluster head 15

3.1 Modelo de dissipação de energia 21
3.2 Exemplo da arquitetura proposta para o PARST com $H = 3$ 22
3.3 Interação dos problemas mestre e subproblema no algoritmo de geração de colunas 29

4.1 Exemplo do método *H-passos*, com $H = 2$ 32
4.2 Comparação entre o Prim e o Prim-H, considerando $H = 3$ 33
4.3 Problema encontrado pelo *Prim-H* para um RSSF com $H = 2$, deixando a rede desconectada, mesmo havendo uma solução. 34
4.4 Heurística construtiva para construir a rota do agente móvel 35
4.5 Exemplo de mal funcionamento da heurística 35
4.6 Uma possível representação para o cromossomo do Algorítmo Genético . . 36
4.7 Árvore de comunicação codificada no cromossomo da Figura 4.6 37
4.8 Representação de um cromossomo. 37
4.9 Exemplo do operador de cruzamento dois pontos 38
4.10 Operador de Mutação 39
4.11 Início da Busca Local com lista 40
4.12 Continuação da Busca Local com lista 41
4.13 Busca local BL_{RC} aplicada a uma rede de comunicação 42
4.14 Exemplo de uma *troca de 2 elementos* na busca 2-Opt 43

5.1 Resultado da análise de variância para diferentes configurações do GRASP 48
5.2 Gráfico da energia total gasta em cada instância pela solução encontrada pelo Algoritmo Genético 53
5.3 Soluções encontrada pelo GRASP para a instância com 100 sensores para os diferentes valores de T_{max} 55
5.4 Comparação entre as soluções obtidas pelo Algoritmo Genético e pelo GRASP 56

5.5 Solução heurística do subproblema incorporada no método de geração de colunas 59
5.6 Análise de variância para diferentes tempos limites para o SUB_{RSSF} em conjunto com o Algoritmo Genético 60
5.7 Análise de variância para diferentes tempos limites para o SUB_{RSSF} em conjunto com o GRASP 60
5.8 Número de *rounds* ao se minimizar a energia total consumida e ao se maximizar o número de rouds utilizando o GRASP como heurística para o subproblema 62
5.9 Comparação para o número de *rounds* e o tempo da geração de colunas considerando o Algoritmo Genético e o GRASP 63
5.10 Comparação do valor da solução com o tempo de execução para a instância `N20-T300` 64

Lista de Tabelas

5.1 Dados dos parâmetros relacionados e seus valores 47
5.2 Parâmetros do algoritmo genético e seus valores 47
5.3 Parâmetros do GRASP e seus valores 47
5.4 Comparação entre os métodos heurísticos *H-passos* e *Prim-H* 48
5.5 Resultado do Algoritmo Exato 49
5.6 Desempenho do Algoritmo Genético proposto para diferentes cenários . . . 52
5.7 Desempenho do GRASP proposto para diferentes cenários 54
5.8 Maximizando o tempo de vida da RSSF com Algoritmo Genético 61
5.9 Maximizando o tempo de vida da RSSF com o GRASP 62

Lista de Abreviaturas

AG	Algoritmo Genético
AM	Agente Móvel
BL	Busca Local
CH	Cluster Head
EB	Estação Base
EECCRS	Energy Efficient Concentric Clustering Routing Scheme
GRASP	Greedy Randomized Adaptive Search Procedure
LC	Lista de candidatos
LEACH	Low-Energy Adaptive Clustering Hierarchy
LEACH-C	LEACH centralizado
LRC	Lista restrita de candidatos
MCFP	Minimum Cost Hop-and-root Constrained Forest Problem
MHS-λ	Multi-Hop Strategy
PARST	Problema Integrado de Agrupamento e Roteamento com Restrição de Salto e Tempo
PCD	Problema do Controle de Densidade
PEGASIS	Power efficient gathering in sensor information systems
PLI	Programação Linear Inteira
PLIM	Programação Linear Inteira Mista
QoS	Qualidade de Serviço
RSSF	Redes de Sensores Sem Fio
SHS	Single Hop Strategy
SPIN	Sensor protocolos for information via negotiation

Resumo

ROMÃO, Oberlan Christo, M.Sc., Universidade Federal de Viçosa, Julho de 2013. **Métodos para redução integrada do consumo de energia e do atraso na entrega de dados em Redes de Sensores Sem Fio**. Orientador: André Gustavo dos Santos

Rede de Sensores Sem Fio (RSSF) surgiram como um campo de pesquisa atraente e desafiador. Um dos principais desafios de tais redes reside nos recursos energéticos limitados disponíveis para nós sensores, uma vez que os sensores são geralmente implantados em ambientes de difícil acesso e em grandes quantidades tornando complicado, ou mesmo impossível, substituir ou recarregar as baterias. Uma possível solução para economizar energia é permitir que um agente móvel percorra a RSSF coletando os dados, mas esta abordagem aumenta o atraso na entrega dos dados. Neste trabalho é usada uma floresta de comunicação, onde as raízes (*cluster heads*) das árvores são os nós sensores visitados pelo agente móvel; os outros sensores enviam seus dados para os *cluster heads* usando um ou mais saltos. Permitir saltos pode diminuir a qualidade de serviço da rede e aumentar o número de falhas, por isso limita-se o número de saltos a um inteiro H. Para controlar o atraso na entrega dos dados, o tempo da trajetória do agente móvel é limitado. Então, o problema é definir os *cluster heads*, a floresta de comunicação com H saltos e a trajetória restrita do agente móvel, minimizando a energia consumida total. É apresentado um modelo de Programação Linear Inteira Mista (PLIM) para o problema definido como *Problema Integrado de Agrupamento e Roteamento com Restrição de Salto e Tempo* (PARST). Como o PLIM se mostrou computacionalmente difícil de se resolver, são propostos métodos híbridos (Algoritmo Genético e GRASP) que definem o conjunto de *cluster heads* usando heurísticas especiais para construir e avaliar as soluções. Uma formulação baseada em geração de colunas também é proposta com o objetivo de aumentar o tempo de vida útil da rede. Resultados são apresentados para a RSSF com até 100 nós sensores usando diferentes limites para o tempo de percurso do agente móvel. A otimalidade das soluções para algumas instâncias com 20 e 30 nós foi confirmada através da resolução da formulação exata do modelo PLIM proposto.

Abstract

ROMÃO, Oberlan Christo, M.Sc., Universidade Federal de Viçosa, July, 2013. **Methods for integrated reducing of energy consumption and delay in the delivery of data in Wireless Sensor Networks**. Adviser: André Gustavo dos Santos

Wireless Sensor Networks (WSNs) have emerged as an attractive and challenging research field. One of the main challenges in such networks lies in the constrained energy resources available to sensor nodes. Since the sensors are usually deployed in hostile environments and in large quantities, it is difficult or impossible to replace or recharge their batteries. A possible solution to save energy is to allow a mobile agent to move through the WSN to collect the data, but this approach increases the delay delivery of messages. In this work a communication forest is used, where the roots (cluster heads) of the trees are the sensors visited by the mobile agent; the other sensors send their information to the cluster heads using one or more hops. Allowing hops can decrease the quality of network service and increase the number of failures, so the number of hops is limited in H. To control the delay data delivery, the time of the mobile agent trajectory is limited. Then, the problem is to define the cluster heads, the communication forest within H hops and the constrained mobile agent path in order to minimize the total energy consumption. It is presented a Mixed-Integer Linear Programming (MILP) formulation for the problem defined as *Integrated Problem of Clustering and Routing with Hop and Time Constrained* (PCRHT). As the MILP showed up computationally hard to solve, hybrid methods (Genetic Algorithm and GRASP) are proposed. These methods define the set of cluster heads using specialized heuristics to build and evaluate the solutions. A formulation based on column generation is also proposed with the aim of increasing the lifetime of the network. Results are presented for WSN with up to 100 nodes sensors using different limits for the travel time of the mobile agent. The optimality of the solutions for some instances with 20 and 30 nodes were confirmed by solving the MILP formulation.

Capítulo 1

Introdução

Uma Rede de Sensores Sem Fio (RSSF) é um tipo especial de rede sem fio *ad hoc* formada por um conjunto de nós sensores e um ou mais nós sorvedouros (também conhecidos como *sink*). Um nó sorvedouro é um nó especial, que pode ser fixo ou móvel, responsável por coletar os dados monitorados pelos nós sensores e transmiti-los para fora da RSSF, e que, por ser especial, é considerado um dispositivo sem restrição de energia, ou seja, com energia inesgotável. O desenvolvimento de redes de sensores sem fio foi motivado por aplicações militares, como a vigilância do campo de batalha; hoje RSSF são utilizadas em muitas aplicações industriais e de consumo, tais como monitoramento de processos industriais e de controle, detecção e prevenção de incêndio, monitoramento de áreas de difícil acesso, monitoramento de ruído e tráfego, e assim por diante. As tarefas realizadas por uma RSSF são feitas de forma colaborativa, ou seja, as informações são disseminadas para outros nós e para um observador final através de pontos de acesso (rádio-base ou nós sorvedouros).

Cada nó sensor da rede é um pequeno dispositivo que, geralmente, é composto por: transmissor de rádio, um processador, memória, placa de sensoriamento e bateria. Com isso, um nó sensor pode realizar o sensoriamento, respondendo a sinais ou estímulos, processamento e comunicação. Por possuírem dimensões e custo reduzido, os nós sensores possuem severas restrições de vários recursos como, por exemplo, energia, largura de banda, capacidade de processamento e armazenamento. Tais restrições, aliadas ao fato de que RSSFs são um novo tipo de rede com particularidades próprias, introduzem novos desafios para o projeto e operação das RSSFs. Dadas suas especificidades, protocolos e algoritmos desenvolvidos para outros tipos de redes são difíceis de serem adaptados às RSSFs (Bechelane *et al.*, 2009). Devido à baixa capacidade de bateria, segundo Akyildiz *et al.* (2002), a economia de energia se torna um dos principais focos de estudo em aplicações em RSSFs que, muitas vezes, estão localizadas em áreas de difícil acesso, tornando complicado, ou mesmo impossível, a reposição de um nó danificado ou de uma bateria esgotada.

Os recentes avanços em sistemas micro eletrônicos (MEMS – Micro Electro Mechanical Systems), comunicação sem fio e eletrônica digital tem estimulado o desenvolvimento de nós sensores de baixo custo que compõem uma Rede de Sensores Sem Fio. Segundo

Valle (2009), a crescente redução no tamanho dos componentes do sensor, aumento da capacidade de processamento e armazenamento de dados, e desenvolvimento de baterias capazes de fornecer uma quantidade cada vez maior de energia para os dispositivos dos sensores permitem que as RSSFs se apresentem como uma solução para diversas aplicações de monitoramento e controle.

Em uma aplicação em RSSF, os nós sensores são depositados em uma área que se deseja monitorar o comportamento de um determinado fenômeno e disseminam as informações coletadas para outros nós e eventualmente para um observador, como ilustrado pela Figura 1.1. Cada nó sensor é responsável pelo monitoramento e coleta das informações de um ou mais fenômenos de interesse. As informações coletadas pelos sensores formam pacotes que são transmitidos ou diretamente ou pelos nós sensores vizinhos até chegar ao nó sorvedouro (também chamado de *sink*). Esta comunicação entre os nós é realizada até que o *sink* receba as informações. O *sink* serve de interface entre a rede e a estação base. Este nó é capaz de se comunicar com a estação base através de um link de comunicação, como, por exemplo, a Internet ou de uma conexão por satélite. Com os dados coletados pela RSSF, o observador pode analisar e tomar decisões.

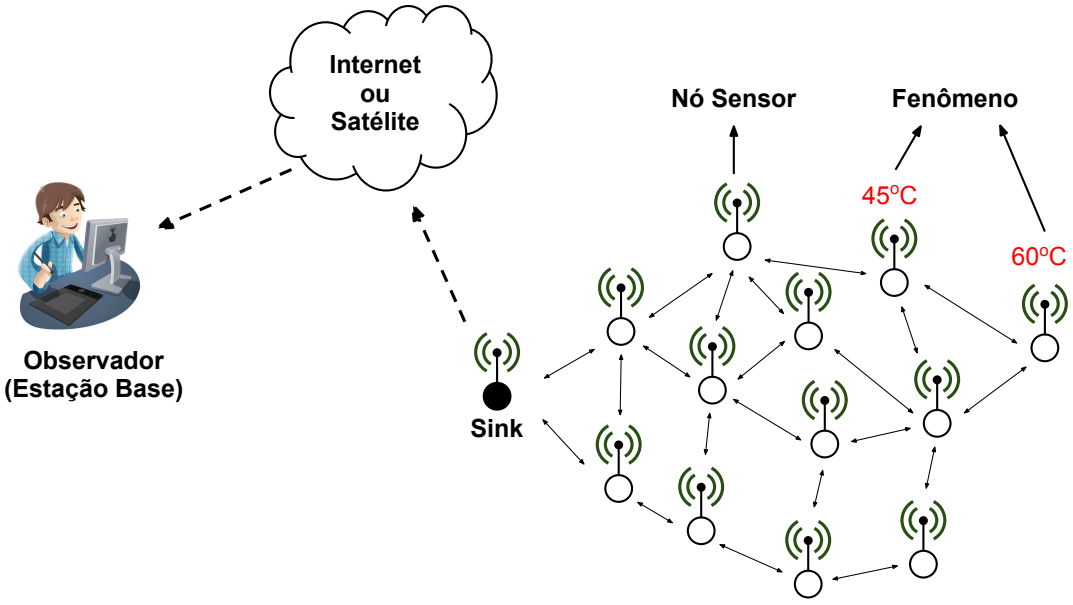


Figura 1.1: Rede de Sensor Sem Fio (Arduino, 2013).

Apesar das RSSFs serem classificadas como um tipo especial de redes *ad hoc*, existem algumas diferenças. Normalmente, RSSFs possuem um grande número e alta densidade de nós distribuídos; nós com restrições de energia, processamento e memória; devem possuir mecanismos para auto-configuração e adaptação devido às falhas; comunicação *broadcast* em contraste com a comunicação ponto-a-ponto das redes tradicionais. Outra característica das RSSFs é sua dependência das aplicações, uma vez que elas determinam o tipo de dispositivo sensor a ser utilizado, a infraestrutura da rede e os requisitos de qualidade de serviço a serem considerados, como a área de cobertura, confiabilidade,

conectividade e o tempo de vida. Dessa forma, os desafios e considerações de projeto de RSSF vão muito além das redes *ad hoc* tradicionais.

Em uma RSSF composta por sorvedouro fixo, os nós sensores podem não conseguir enviar seus dados coletados diretamente ao sorvedouro, devido o seu raio de comunicação limitado. Dessa forma, outros nós sensores deverão ser empregados para rotear os dados ao seu destino (multi-saltos). Entretanto, segundo Kim *et al.* (2003), a comunicação em RSSF usando multi-saltos é o principal responsável pelo consumo de energia da rede. Assim, uma estratégia que tem sido adotada para reduzir o consumo de energia da rede consiste em limitar o número de saltos. Entretanto, usar essa abordagem pode tornar a rede desconectada, uma vez que alguns nós sensores podem não ser capazes de se comunicarem com outros sensores devido a restrição de saltos e o limite de comunicação entre os nós sensores. Assim, o uso de sorvedouro móvel (agente móvel) se torna uma estratégia capaz de tratar este problema. Uma abordagem muito usada é agrupar os nós sensores em pequenos conjuntos e um nó, de cada conjunto, é eleito como líder do grupo (*cluster head*), que tem a função de receber os dados dos outros nós do conjunto, e em seguida transmiti-los ao agente móvel, que visita estes nós coletando os dados monitorados.

A substituição da comunicação direta entre os nós sensores pela comunicação sensor e agente móvel acarreta um novo problema. Por terem velocidade de movimentação muito menor que a velocidade de transmissão sem fio, permitir sorvedouro móvel na rede aumenta drasticamente o atraso na entrega dos dados, que é o tempo decorrido entre o momento em que o dado é coletado (gerado) pelo nó sensor e o momento em que ele chega à estação base (Wang *et al.*, 2005). Uma estratégia para diminuir o atraso é limitar a distância ou o tempo que o agente móvel gasta em sua trajetória pela rede. Portanto, a construção e operação de uma RSSF deve ser cuidadosamente planejada, a fim de aumentar o tempo em que a rede é capaz de realizar as suas operações e diminuir o atraso de entrega da mensagem.

1.1 Objetivos

O objetivo geral deste trabalho é propor métodos para redução integrada do consumo de energia e do atraso na entrega de dados em redes de sensores sem fio utilizando agente móvel. Nesse contexto, deve-se encontrar a quantidade e o melhor agrupamento para os nós sensores, definir o *cluster head* de cada agrupamento, construir a rede de comunicação com um número de saltos restrito, e encontrar a rota limitada de menor caminho (tempo) para o agente móvel, minimizando o consumo de energia total da rede. Para isso, é proposta uma formulação matemática para o problema, aqui denominado, *Problema Integrado de Agrupamento e Roteamento com Restrição de Salto e Tempo* (PARST).

Especificamente, pretende-se:

- Implementar um modelo matemático para o PARST para minimizar a energia total

gasta pela RSSF, considerando as diversas restrições do problema;

- Incorporar um modelo de energia para a RSSF na formulação matemática do PARST;
- Implementar uma abordagem baseada em geração de colunas com o objetivo de aumentar o tempo de vida da rede;
- Propor e aplicar métodos heurísticos para o PARST;
- Propor e aplicar buscas locais paras os métodos heurísticos;
- Comparar o desempenho dos métodos heurísticos propostos com a formulação exata;
- Adaptar os métodos heurísticos para que funcionem como geradores de colunas para a geração de colunas;
- Apresentar resultados que comprovem a viabilidade da utilização dos métodos heurísticos.

1.2 Principais contribuições

As principais contribuições deste trabalho são:

- Modelagem dos problemas de roteamento e agrupamento de forma integrada, restringindo o número de saltos e o tempo de percurso do agente móvel, que minimiza a energia total consumida pela RSSF;
- Definição do modelo de consumo de energia para a aplicação;
- Definição de uma abordagem para aumentar o tempo de vida da rede;
- Desenvolvimento de métodos híbridos que são auxiliados por heurísticas e buscas locais para avaliar e construir soluções de boa (ou ótima) qualidade para o PARST;
- Extensiva avaliação dos métodos propostos, estudando o impacto no consumo total de energia e no tempo de vida da RSSF ao se limitar o tempo de percurso do agente móvel.

1.3 Organização da dissertação

Esta dissertação está dividida em seis capítulos, incluindo esta introdução. No Capítulo 2, são apresentados os principais problemas em RSSFs tratados neste trabalho, bem como uma revisão bibliográfica dos mesmos. No Capítulo 3, é introduzido o PARST, é apresentada uma formulação matemática para o mesmo e uma abordagem usando geração de colunas para aumentar o tempo de vida da rede.

No Capítulo 4, métodos híbridos desenvolvidos, bem como as heurísticas construtivas e as buscas locais, são apresentados. No Capítulo 5, são apresentados os resultados obtidos, tanto pela formulação exata quanto pelo método híbrido, e sua análise.

Finalmente, encerra-se a dissertação no Capítulo 6, com as principais conclusões do trabalho e as direções de pesquisas futuras.

Capítulo 2

Rede de Sensores Sem Fio

Neste capítulo, são apresentados os principais conceitos envolvidos na organização de Rede de Sensores Sem Fio. São apresentadas aplicações práticas e específicas que justificam o interesse em pesquisas na área e restrições que diferenciam as RSSFs das redes sem fio tradicionais. Em seguida, é feita uma revisão bibliográfica de diversos problemas comumente abordados em RSSFs. Conclui-se o capítulo apresentando as contribuições desta dissertação.

2.1 Conceitos e Aplicações

Redes de Sensores Sem Fio (RSSFs) são um tipo de rede *Ad-hoc* que geralmente consistem de um grande número de nós sensores que são colocados em um ambiente do qual se deseja obter informações (Zheng & Jamalipour, 2009). Cada nó sensor é um pequeno dispositivo que geralmente possui uma placa de sensoriamento responsável por coletar os dados, um microprocessador com poder de processamento limitado, uma pequena memória, um radio comunicador para realizar a comunicação com outros sensores, e uma bateria com energia limitada. Portanto, além de terem a capacidade de monitorar o ambiente, os sensores também podem processar os dados sensoriados e se comunicar a uma curta distância via *wireless*. A Figura 2.1 mostra um modelo de um sensor MICA2.

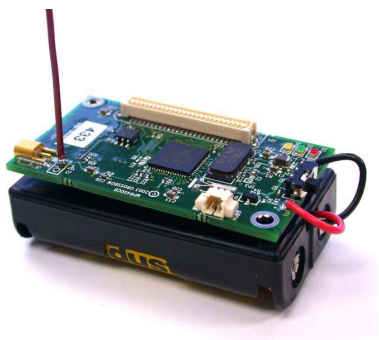


Figura 2.1: Modelo de nó sensor MICA2 (XBOW, 2013).

Todas as operações realizadas pelos nós das RSSFs devem ser analisadas quanto ao uso de memória, processamento e principalmente em relação ao consumo de energia. Redes de sensores sem fio diferem de redes de computadores tradicionais em vários aspectos, principalmente por possuírem um grande número de nós sensores distribuídos com severas restrições de energia, uma vez que os sensores são geralmente distribuídos em ambientes de difícil acesso e em grandes quantidades, o que dificulta ou impossibilita a troca ou recarga de suas baterias. Assim, qualquer estudo relacionado as RSSFs deve considerar o consumo de energia como o principal requisito, já que o tempo de vida do nó sensor depende da quantidade de energia disponível. Dessa forma, estender o tempo de vida da rede é um dos principais focos das pesquisas em RSSFs.

Segundo Valle (2009), a crescente redução no tamanho dos componentes que compõem o sensor, o aumento da capacidade de processamento e armazenamento de dados, e o desenvolvimento de baterias capazes de fornecer cada vez mais energia para os sensores permitem que as RSSFs se apresentem como uma solução para diversas aplicações de monitoração e controle, tais como: monitoramento ambiental, monitoramento e controle industrial, segurança pública e de ambientes em geral, áreas de desastres e de riscos para vidas humanas, transporte e controle militar. Dessa forma, vários tipos de eventos, fenômenos e propriedades podem ser monitorada tais como: temperatura, radiação, atividade eólica, umidade, movimento, detecção de incêndio, pressão, qualidade do ar ou água, nível de ruído, presença ou ausência de objetos, entre outros (Akyildiz *et al.*, 2002). A Figura 2.2 mostra um exemplo de uma RSSF aplicada no campo militar. Os sensores, presentes nos veículos e na tropa, detectam possíveis ameaças (bombas terrestres ou inimigos camuflados) e se comunicam entre si, enviando os dados coletados até uma *estação base (EB)*, onde as informações coletadas pela RSSF são processadas.

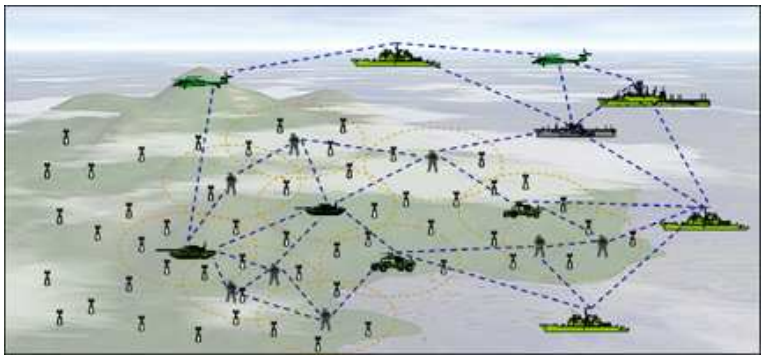


Figura 2.2: Exemplo de aplicação de uma RSSF para o controle militar (ECE, 2013).

Observe que as aplicações das RSSFs são bastantes distintas entre si e, por se tratar de uma rede especial, cada RSSF possui sua particularidade, relacionada à sua função de aplicação e seus objetivos. Assim, o projeto de uma RSSF depende de sua aplicação, podendo ser influenciado por diversos fatores e pré-requisitos, tais como: a necessidade de tolerância a falhas, os custos de produção, o ambiente operacional, qualidade de serviço,

garantia da conectividade entre os nós sensores ativos, o roteamento dos dados coletados e o consumo de energia (Bechelane, 2009; Valle, 2009). Dessa forma, o protocolo de roteamento e agrupamento deve ser baseado nas características de cada rede. Segundo (Brittes, 2007), algumas RSSFs têm necessidade de monitoramento e transferência constante de dados, outras podem obter os dados com um maior intervalo de tempo, e existem ainda as que somente obtêm dados quando solicitados pelo usuário observador junto à estação base. Portanto, é possível empregar técnicas específicas de roteamento e agrupamento considerando as limitações da RSSF, com o objetivo de minimizar problemas potenciais que podem ocorrer durante a coleta e transmissão dos dados.

De acordo com (Brittes, 2007), as principais características a serem analisadas em uma RSSF e que influenciam na escolha dos protocolos de roteamento são:

- *Recursos limitados*: a capacidade energética e de processamento de uma RSSF são limitadas e o alcance de transmissão dos sensores é restrito;
- *Topologia dinâmica*: os protocolos devem considerar a mobilidade dos sensores. Pode haver falha de nós e para isso os protocolos devem considerar redundância de informações para garantir o bom funcionamento da rede. Os sensores podem ficar inativos em períodos de baixa atividade a fim de economizar energia;
- *Tempo de vida da rede*: o tempo de vida da RSSF deve ser o maior possível para minimizar o custo de manutenção da rede.

Em RSSF pode existir um nó especial denominado *sorvedouro* ou *sink*, que tem a função de recolher os dados coletados e analisados pelos nós sensores da RSSF, e enviá-los à estação base. Uma RSSF pode ter um ou mais nós sorvedouros, que podem ser fixos ou móveis. Normalmente, o nó sorvedouro não possui restrição de energia, ou seja, considera-se que estes possuem energia ilimitada.

Neste trabalho, assume-se que a RSSF possui um nó sorvedouro móvel, denominado *Agente Móvel (AM)*, que sobrevoa a área de interesse. Além disso, considera-se que o agente móvel se movimenta em linha reta de uma posição para outra, com velocidade constante, e possui energia e espaço de armazenamento ilimitado.

Enquanto as tradicionais redes *ad-hoc* tem o objetivo de obter um alto índice de *Qualidade de Serviço* (QoS, do inglês *Quality of Sevice*), as RSSFs procuram reduzir o consumo de energia em suas operações. Segundo Valle (2009), é comum existir um compromisso entre o tempo de vida útil da rede (*lifetime*) com outras métricas importantes, como o atraso na entrega das mensagens. O tempo de vida da rede pode ser definido como o tempo decorrido até o primeiro sensor na rede deixar de funcionar (Keskin *et al.*, 2011; Yun & Xia, 2010; Behdani *et al.*, 2012); e o atraso na entrega das mensagens é o tempo gasto entre a geração de uma informação pelo sensor e o momento que o dado é recebido pela estação base (Valle, 2009).

Dessa forma, várias abordagens têm sido desenvolvidas para reduzir o consumo de energia da rede e, assim, aumentar seu tempo de vida. Dentre estas abordagens, pode-se citar estratégias para disseminação mais eficiente dos dados e controle de densidade, explicados a seguir. Além destas estratégias, muitos trabalhos consideram a mobilidade do nó sorvedouro para diminuir a energia gasta no roteamento dos dados entre o nó sensor e a estação base.

2.2 Trabalhos da literatura em RSSF

2.2.1 Disseminação da Informação

A definição da topologia em uma rede de sensores sem fio envolve estabelecer como a informação será disseminada entre os nós e o sorvedouro e/ou estação base, uma vez que a comunicação é o procedimento que mais consome energia (Akyildiz *et al.*, 2002; Kim *et al.*, 2003).

Segundo Al-Karaki & Kamal (2004), uma definição importante na estratégia de disseminação de informação em RSSF é o número de saltos no caminho da transmissão da informação entre o nó sensor e o sorvedouro. Nesse sentido, quando os dados de cada sensor são enviados diretamente para o sorvedouro, a RSSF é classificada como *single-hop* (um salto). Quando é permitida a retransmissão dos dados, a RSSF é classificada como *multi-hop* (múltiplos saltos).

Na abordagem *single-hop* é comum encontrar o método de roteamento utilizando o conceito de *clusters* (agrupamento). Nesta técnica, os nós enviam os seus dados diretamente (com um único salto) para o líder do grupo (do inglês *Cluster Head*), responsável por repassar os dados do seu *cluster* para a estação base, o que limita o tamanho da rede em função do raio de alcance de comunicação dos sensores. Além disso, os *cluster heads* esgotam sua energia rapidamente, já que são os responsáveis por transmitir os dados do grupo a uma distância relativamente grande. Em redes esparsas, alguns nós sensores podem ficar inacessíveis devido ao raio limitado de comunicação dos nós sensores, tornando a rede desconectada. Um dos protocolos baseados em agrupamento *single-hop* mais conhecidos é o LEACH (*Low-Energy Adaptive Clustering Hierarchy*) proposto por Heinzelman *et al.* (2000). A Figura 2.3 mostra um exemplo de topologia obtida a partir do protocolo LEACH, onde os sensores são agrupados e em cada grupo um nó sensor é eleito como *cluster head*. Os sensores enviam seus dados diretamente para o *cluster head* do seu grupo, que por sua vez, envia os dados do grupo diretamente para a estação base. Essa abordagem pode ser encontrada, entre outros, em Heidari & Movaghar (2011), Seo *et al.* (2009) e Heinzelman *et al.* (2002).

Heinzelman *et al.* (2002) propuseram uma versão do LEACH centralizado (LEACH-C). Ao contrário do LEACH, onde os nós se auto-configuram em *clusters*, o LEACH-C utiliza

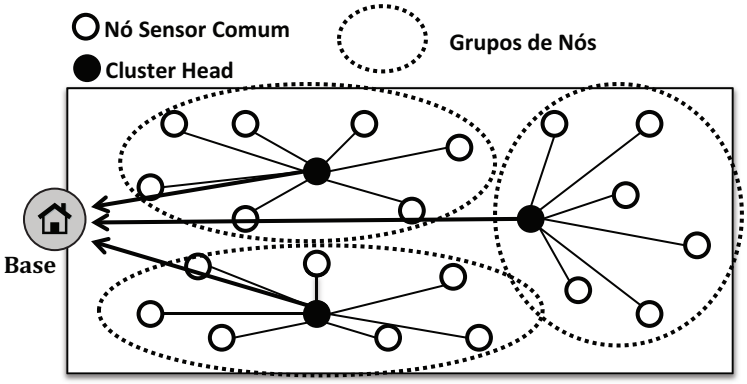


Figura 2.3: Topologia de uma RSSF baseada em agrupamentos (Brittes, 2007).

as informações da estação base para formar os *clusters*. Durante a fase de configuração de LEACH-C, a estação base recebe a informação sobre a localização e nível de energia de cada nó da rede, e com essa informação, a estação base encontra um número pré-determinado de *cluster heads* e configura a rede em *clusters* para a próxima rodada, que pode ser por um determinado tempo ou vezes que os dados serão coletados. Além disso, sensores que possuem energia abaixo da média da rede não podem ser *cluster heads* na rodada atual. Embora as outras operações de LEACH-C sejam idênticas as do LEACH, os resultados apresentados em Heinzelman *et al.* (2002), indicam uma melhoria significativa quanto a economia de energia comparado ao LEACH.

Por outro lado, na abordagem *multi-hop*, os pacotes são enviados de um nó para outro em saltos, até chegarem à estação base ou ao sorvedouro, como mostra a Figura 2.4. Nesse caso, para se calcular a menor distância, pode-se usar protocolos baseados no roteamento para redes cabeadas: Vetor de Distância (envia os pacotes na direção genérica de seu destino) ou em Estado de enlace (mantêm uma tabela completa contendo os caminhos de cada nó). Entretanto, o tráfego de mensagens e a manutenção de rotas válidas congestionam a rede e consomem muita energia (restrita) das baterias (Brittes, 2007).

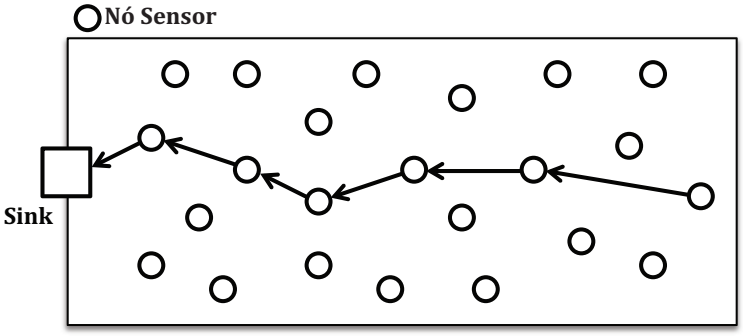


Figura 2.4: Mensagem percorrendo RSSF em esquema *multi-hop* (Brittes, 2007).

Dentre os protocolos multi-saltos para RSSF, o SPIN (*Sensor Protocol for Information via Negotiation*), apresentado por Heinzelman *et al.* (1999), é o mais antigo (Akkaya & Younis, 2005). Ele é um protocolo que usa informações sobre o nível de energia disponível

em cada nó para fazer o roteamento dos dados e usa protocolos de negociação para eliminar a transmissão redundante de dados na rede. Além disso, para diminuir o gasto energético com a transmissão dos dados, sensores com pouca energia participam menos das transmissões de dados.

Uma técnica também muito utilizada na transmissão de dados é a inundação, do inglês *flooding*, que pode ser utilizada em RSSF. Cada sensor recebe um pacote de dados e o transmite a todos os seus vizinhos. E este processo continua até que o pacote seja recebido pelo destinatário ou seja atingido o número máximo de saltos estabelecido (Akkaya & Younis, 2005). Embora o *flooding* seja de fácil implementação, ele possui alguns problemas, como a implosão (mensagens duplicadas são enviadas ao mesmo nó) e a ineficiente utilização dos recursos energéticos da rede. Para contornar estes problemas foi desenvolvido o *gossiping* (Akkaya & Younis, 2005), onde o nó envia seus dados para um vizinho aleatório. Entretanto, por este algoritmo o tempo para a propagação da mensagem pode ser longo demais.

Lindsey & Raghavendra (2002) propuseram o PEGASIS (*Power efficient gathering in sensor information systems*), que é um protocolo de comunicação baseado no LEACH, mas que permite multi-saltos. No PEGASIS, um conjunto de cadeias de nós é formada para a transmissão dos dados até o sorvedouro e todos os nós processam a agregação de dados. Apenas um nó transmite à estação base por rodada. Além disso, cada sensor pode se comunicar apenas com o sensor mais próximo. Já Park *et al.* (2009) propuseram o EECCRS (*Energy Efficient Concentric Clustering Routing Scheme*), que é baseado no PEGASIS, mas cada cadeia possui a mesma quantidade de sensores, balanceando a dissipação de energia.

2.2.2 Controle de Densidade

Em uma RSSF podem existir vários nós sensores cobrindo (sensoriando) uma mesma região, o que gera um desperdício de energia, redundância de dados e aumento tráfego da rede. Por isso, em algumas pesquisas, como em Slijepcevic & Potkonjak (2001); Cardei *et al.* (2002); Meguerdichian & Potkonjak (2003); Nakamura *et al.* (2005); Siqueira *et al.* (2006) e Aioffi (2007), implementa-se um controle de densidade na RSSF com o objetivo de diminuir a área de redundância da rede e, consequentemente, aumentar sua vida útil, uma vez que muitos nós poderiam ser desligados sem que alguma área de interesse deixasse de ser monitorada. Para que isso aconteça, alguns nós sensores são programados para dormir, enquanto outros continuam os serviços de monitoramento, coleta, processamento e transmissão de dados. Dessa forma, os nós que estão ativos são alternados diminuindo o gasto de energia em cada iteração e, consequentemente, aumentando o tempo de vida da RSSF.

No trabalho de Nakamura *et al.* (2005) é proposto o Problema do Controle de Densidade (PCD), que define o subconjunto de nós que devem ficar ativos visando minimizar o

consumo de energia na rede. O PCD garante a cobertura, conectividade e o roteamento em uma RSSF no intuito de minimizar o consumo de energia total da rede fornecendo uma solução onde se obtém, para cada intervalo de tempo, a melhor topologia da rede, ao definir quais nós estarão ativos e a rota entre estes nós e o sorvedouro ou estação base.

Huang & Tseng (2003) definem o problema de cobertura para RSSF como uma determinação de quão bem a região é monitorada ou controlada pelos sensores. Os autores formulam o problema como um problema de decisão com o objetivo de determinar se pelo menos k sensores cobrem cada ponto da área de interesse. Já Meguerdichian & Potkonjak (2003) apresentam modelos matemáticos de Programação Linear Inteira (PLI) para resolver o problema de cobertura em rede de sensores, controlando a densidade de nós ativos na rede.

No trabalho de Slijepcevic & Potkonjak (2001) é apresentada uma heurística centralizada para dividir o conjunto de nós sensores da rede em sub-conjuntos, onde cada sub-conjunto de nós sensores cobre totalmente a região de interesse. O objetivo da heurística é maximizar o número n de sub-conjuntos, aumentando o tempo de vida da rede em até n vezes.

2.2.3 Mobilidade do Sorvedouro

A utilização de sorvedouros fixos é uma abordagem que implica em alto consumo de energia. Assim, é natural considerar a mobilidade dos sorvedouros como uma alternativa. Além de reduzir o consumo de energia, estendendo o tempo de vida da rede, a utilização de sorvedouros móveis (também conhecidos como agentes móveis) também permite que redes esparsas sejam conectadas. Entretanto, como a velocidade do agente móvel é muito menor que a velocidade de transmissão dos dados entre os nós sensores, esta abordagem aumenta drasticamente o atraso na entrega dos dados (Valle, 2009; Wang *et al.*, 2005). Algumas pesquisas na área de robótica podem oferecer suporte ao requisito de mobilidade do sorvedouro. Um exemplo é o projeto *WiFly* (Wifly, 2013), onde um nó sensor é acoplado a um avião de pequena dimensão controlado remotamente, como ilustrado na Figura 2.5.

Figura 2.5: Avião do projeto WiFly (Wifly, 2013).

Como ilustrado anteriormente, vários trabalhos da literatura propõem redes sem o

uso de agentes móveis permitindo multi-saltos na comunicação entre os sensores. Neste caso, os próprios sensores são responsáveis por retransmitir as informações sensoriadas pela rede até a estação base. Entretanto, segundo Kim *et al.* (2003), a transmissão de dados na RSSF com multi-saltos é a principal responsável pelo consumo de energia da rede. Portanto, muitos trabalhos da literatura, como Kim *et al.* (2003); Wang *et al.* (2005); Aioffi *et al.* (2007); Bechelane *et al.* (2009); Yun & Xia (2010); Cai *et al.* (2011); Keskin *et al.* (2011); Behdani *et al.* (2012), propõem a comunicação entre os sensores usando um número controlado (limitado) de saltos. Neste caso, o uso de um agente móvel se torna necessário, uma vez que alguns sensores podem não ser capazes de enviar seus dados usando um número limitado de saltos.

Keskin *et al.* (2011) propõem duas formulações de programação matemática: uma limitando o número de saltos e outra sem limite. No trabalho, é definido um pequeno conjunto de pontos que o agente móvel pode visitar e os nós sensores devem enviar seus dados coletados para esses pontos. Song & Hatzinakos (2007) apresentam uma arquitetura para um aplicativo de vigilância de tráfego, onde um carro de polícia é o agente móvel que percorre uma estrada para a coleta dos dados. Nesta aplicação, é considerado que todos os sensores devem se comunicar com o agente móvel utilizando um salto.

Heurísticas baseadas em algoritmos genéticos foram propostos, entre outros, por Cai *et al.* (2011) e Wu *et al.* (2004) para o planejamento de itinerário do agente móvel. Enquanto Wu *et al.* (2004) considera apenas um sorvedouro móvel, Cai *et al.* (2011) permite múltiplos agente móveis. Em ambos os trabalhos, saltos não são permitidos e os dados de cada sensor devem ser enviados diretamente para algum agente móvel, ou seja, o(s) agente(s) móvel(is) deve(m) visitar todos os sensores. Isto pode resultar em uma alta latência, ou seja, um grande atraso na entrega dos dados coletados, mesmo usando mais de um agente móvel.

Aioffi (2007) propõe duas técnicas de agrupamento dos sensores para facilitar a coleta de dados pelo agente móvel: SHS (*Single Hop Strategy*) e o MHS-λ (*Multi-Hop Strategy*). No SHS, os nós sensores são agrupados e enviam seus dados ao agente móvel diretamente (*single-hop*), que fica por alguns instantes no centro do agrupamento. O objetivo do SHS é definir o menor número de agrupamentos, o que diminui o tempo da trajetória do agente móvel e, consequentemente, diminui o atraso na entrega dos dados. O SHS é resolvido em duas fases, separadamente. A primeira consiste em encontrar o menor número de agrupamentos possível, respeitando a restrição de raio de comunicação dos nós sensores. Em seguida, é resolvido o problema de roteamento, com o objetivo de encontrar a menor trajetória do agente móvel, que deve visitar o centro de cada agrupamento. Entretanto, como os sensores só podem enviar seus dados quando o agente móvel estiver próximo e geralmente o número de agrupamentos é grande, o atraso na entrega dos dados ainda pode ser alto. A Figura 2.6 ilustra a topologia SHS aplicado a RSSF.

Por outro lado, o MHS-λ permite comunicação multi-saltos, limitando o número de saltos em λ. Com a comunicação multi-saltos, os agrupamentos se tornam maiores, dimi-

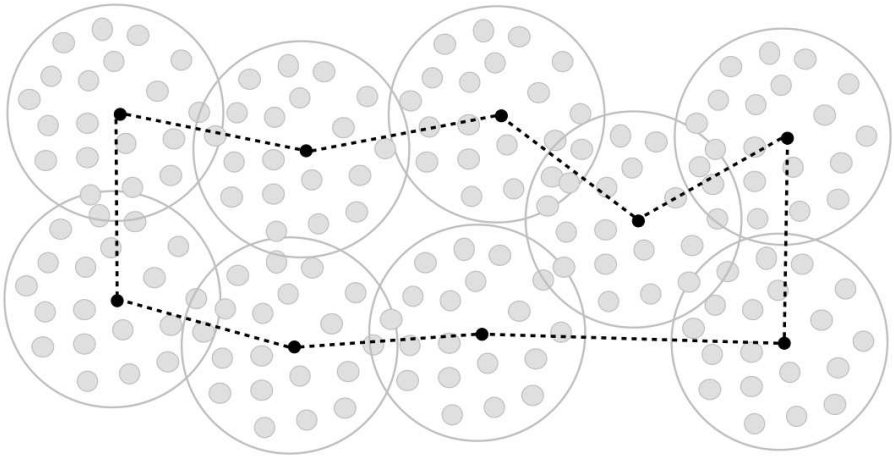


Figura 2.6: Exemplo da topologia da rede SHS (Aioffi, 2007).

nuindo o tempo gasto pelo agente móvel na trajetória, quando comparado com o SHS. Diferentemente do SHS, no MHS-λ, em cada agrupamento é escolhido um *cluster head*, que fica encarregado de enviar os dados para o agente móvel. Dessa forma, o agente móvel deve visitar cada *cluster head*. Similarmente ao SHS, o MHS-λ também é resolvido em duas fases: primeiramente é resolvido o problema de agrupamento; e, em seguida, de forma independente, é resolvido o problema de roteamento. A Figura 2.7 mostra a topologia MHS-3 aplicado a uma RSSF.

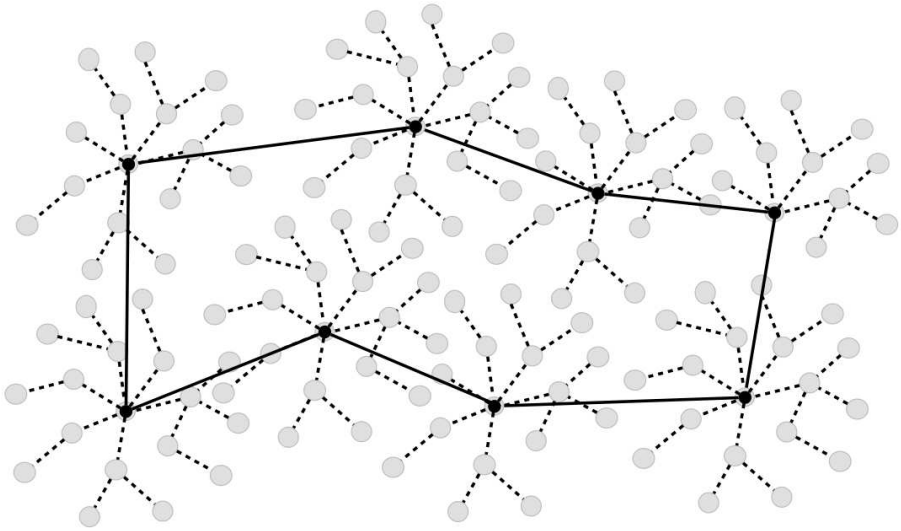


Figura 2.7: Exemplo da topologia da rede MHS-3 (Aioffi, 2007).

O *Minimum Cost Hop-and-root Constrained Forest Problem* (*MCFP*) foi proposto por Bechelane *et al.* (2009), onde é apresentada uma formulação de Programação Linear Inteira Mista (PLIM) que objetiva minimizar a energia gasta pela rede, limitando o tamanho da rota do agente móvel e o número de saltos. Também é proposta uma heurística para auxiliar o PLIM. O MCFP é similar ao MHS-λ, mas o problema de agrupamento e roteamento são resolvidos de forma integrada. Entretanto, a abordagem assume que cada sensor envia apenas um pacote de dados, ou seja, os dados coletados e enviados a outro sensor são

agregados e apenas um pacote é enviado ao próximo nó sensor, que repete a agregação dos dados. Para algumas aplicações, como informar a média da temperatura em uma região, essa abordagem é válida, mas para aplicações onde todas as informações são necessárias, como em aplicações militares, ela se torna inaplicável. Além disso, Bechelane *et al.* (2009) consideram apenas a energia gasta na transmissão de dados, e não no recebimento. Para diminuir o atraso na entrega dos dados, Bechelane (2009) estende o MCFP permitindo múltiplos agentes móveis.

2.3 Contribuições

Como pode ser observado nas seções anteriores, existe uma infinidade de problemas de otimização com objetivos conflitantes em RSSFs. Por isso, uma RSSF precisa englobar mecanismos que permitam balancear o tempo de vida da rede e requisitos específicos de QoS, como o atraso na entrega dos dados.

Neste trabalho é usada uma arquitetura de rede em que os nós sensores são organizados por meio de um conjunto de árvores de comunicação, onde as arestas representam uma comunicação entre dois sensores, e o sorvedouro é móvel. Além disso, existe um limite H para o número de saltos entre qualquer sensor e a raiz da árvore à qual o nó pertence; e um sensor pode se comunicar com outro sensor se estiverem dentro do alcance de comunicação, dado pelo raio (R^c) de comunicação. O agente móvel percorre as raízes das árvores (*cluster heads*) coletando os dados, permanecendo um determinado tempo (T^{ch} segundos) em cada *cluster head*, para que este possa transmitir os seus dados ao agente móvel. Apenas os *cluster heads* podem se comunicar com o agente móvel, que se move em linha reta a uma velocidade constante. Considera-se também que o agente móvel voa a uma altura fixa (H_{AM}) acima dos *cluster heads*, que é a distância de transmissão usada pelos *cluster heads*. A estação base, assim como o agente móvel, é rica em energia (ilimitada) e também atua como *cluster head*. A Figura 2.8 ilustra a arquitetura da RSSF adotada no trabalho, considerando o limite de saltos $H = 3$. Já a Figura 2.9 exemplifica o agente móvel sobrevoando a região sensoriada e recebendo os dados dos *cluster heads*.

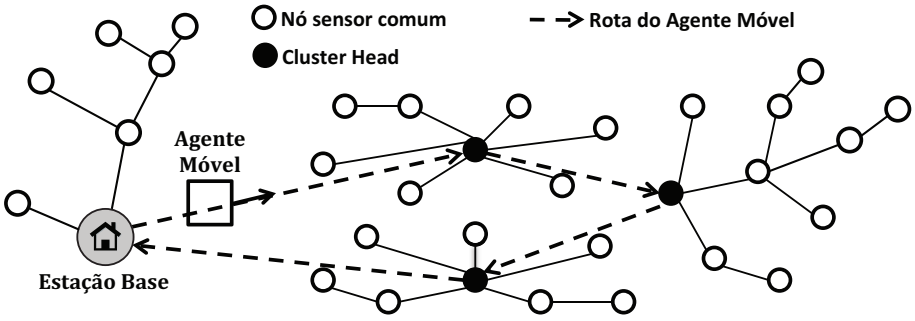


Figura 2.8: Exemplo da topologia adotada com restrição de saltos H = 3.

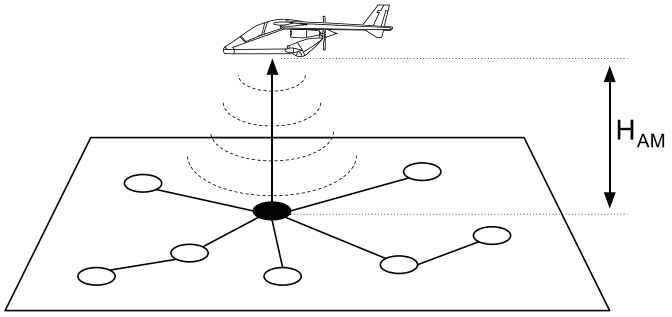


Figura 2.9: Agente Móvel voando, a uma altura H_{AM}, sobre a região sensoriada e recebendo os dados de um cluster head.

O objetivo principal da abordagem proposta neste trabalho é minimizar a energia total gasta na RSSF utilizando agente móvel. Entretanto, em redes que usam agente móvel, minimizar a energia gasta na rede normalmente implica em rotas mais longas para o AM, ou seja, um atraso maior na entrega das informações. Para contornar este problema, limita-se o tempo gasto pelo AM na rota em T_{max}, com o objetivo de ter um equilíbrio entre energia e latência. Dessa forma, o atraso na entrega dos dados pode ser controlado pelo projetista da RSSF.

Este trabalho é uma adaptação do MCFP, proposto por Bechelane *et al.* (2009). Além de se considerar a energia gasta na transmissão e recepção dos dados, é levada em conta a quantidade de informação que é transmitida: se um nó sensor recebe informação de um outro nó, ele não irá processar e empacotar os dados em uma só mensagem, pois em algumas aplicações os dados coletados de diferente sensores não podem ser agregados em um único pacote, ou seja, todas as informação coletadas pela rede devem ser enviadas para estação base. Ao invés de se considerar a distância como um comparativo com o atraso na entrega das informações, como é feito por Bechelane *et al.* (2009), é calculado o tempo gasto pelo agente móvel na rota, que representa uma estimativa mais real ao atraso na entrega dos dados.

A arquitetura proposta também é bastante parecida com a MHS-λ proposta por Aioffi (2007). Entretanto, os problemas de agrupamento dos nós sensores e de roteamento do agente móvel são tratados de forma integrada, diferentemente da abordagem MHS-λ, que trata tais problemas separadamente.

2.4 Comentários Finais

Esse capítulo apresenta alguns conceitos e aplicações de RSSF. Também mostra alguns trabalhos relacionados da literatura, juntamente com o que distingue este do restante.

No próximo capítulo, o problema tratado é definido como problemas em grafos e uma formulação matemática para o mesmo é apresentada. Com o objetivo de aumentar o tempo de vida da rede, também é apresentada uma abordagem com geração de colunas.

Capítulo 3

Problema Integrado de Agrupamento e Roteamento com Restrição de Salto e Tempo (PARST)

Neste capítulo é apresentada uma formulação matemática para o problema de agrupamento e roteamento para uma RSSF. O problema em estudo, denominado *Problema Integrado de Agrupamento e Roteamento com Restrição de Salto e Tempo* (PARST), é uma junção do Problema de Árvore Geradora de Custo Mínimo com Restrição de Saltos (Gouveia, 1996) e o Problema do Caixeiro Viajante (Dantzig *et al.*, 1954). Apresenta-se uma formulação em grafos e uma formulação de Programação Linear Inteira Mista, minimizando a energia total gasta pela rede restringindo o tempo máximo da rota do agente móvel e o número de saltos.

3.1 O Modelo Proposto

O Problema Integrado de Agrupamento e Roteamento com Restrição de Salto e Tempo (PARST) proposto, pode ser descrito em um grafo $D = (V, A)$. O conjunto de vértices $V = \{1, 2, \ldots, n\}$ representa os nós sensores, localizados no plano Euclidiano. O conjunto de arestas A será empregado para modelar a rota do agente móvel e o subconjunto $\widehat{A} \subseteq A$ para a modelagem das árvores de comunicação. Note que $(i,j) \in \widehat{A}$ se os sensores i e j podem se comunicar diretamente, isto é, se eles estiverem dentro do alcance de comunicação.

O conjunto A usado na modelagem da rota do agente móvel é completo, representando todas as possíveis translações dos agente móveis, movendo de um *cluster head* a outro. A distância Euclidiana $\{d_{ij} : i,j \in V, i \neq j\}$ é associada a cada aresta $(i,j) \in A$. A estação base é definida como o vértice $1 \in V$, que também é o ponto de partida e término da rota do agente móvel. Dessa forma, o vértice 1 sempre será *cluster head*, ou seja, a raiz de uma árvore de comunicação.

O agente móvel visita um número l de *cluster heads*, que não é determinado a priori. Seja $C \subseteq V$ o conjunto de todos os *cluster heads*, onde $|C| = l$. Dessa forma, a trajetória do agente móvel consiste em visitar os *cluster heads* $C = \{c_1, \ldots, c_l\}$. A rota do agente móvel deve incluir todos os *cluster heads*, formando um ciclo Hamiltoniano $(c_1, c_2), \ldots, (c_{l-1}, c_l), (c_l, c_1)$ de comprimento total C_{rota}.

Assume-se que o agente móvel se desloca em linha reta a uma velocidade constante de v_{am} m/s e permanece por T^{ch} segundos em cada *cluster head*, para que este possa transmitir os seus dados ao agente móvel. Dessa forma, o agente móvel gasta $T = \frac{C_{rota}}{v_{am}} + l \times T^{ch}$ segundos para percorrer a rota, onde T não deve ser superior a T_{max}.

Para construir as árvores de comunicação, é definido o subconjunto de arestas $\widehat{A} \subseteq A$, considera-se as distâncias Euclidianas $\{d_{ij} : i,j \in V, i \neq j\}$ entre os vértices de V, e também que o raio máximo de comunicação dos nós sensores é dado por um número real positivo R^c. Assim, a aresta $(i, j) \in A$ se e somente se $d_{ij} \leq R^c$.

A floresta $F = (V, A_F)$ de D representa as árvores de comunicação, e consiste em uma coleção de árvores dirigidas disjuntas $T_i = (V_i, A_i), i = 1, \ldots, l$, de forma que $A_F \subseteq \widehat{A}$ e $A_F = \bigcup_{i=1}^{l} A_i$. Os conjuntos de vértices V_i são tais que $\bigcup_{i=1}^{l} V_i = V$, garantindo a cobertura da rede. Dado um número natural H, F é chamada de floresta restrita a H saltos de D se, para cada árvore T_i, o número máximo de arestas existentes no caminho da sua raiz para qualquer $j \in V_i$ não excede H. As raízes das árvores representam os *cluster heads* e são os nós sensores visitados pelo agente móvel.

Note que, se o tempo da rota do agente móvel for limitado em zero, ou seja, $T_{max} = 0$, o PARST se reduz ao Problema da Árvore Geradora Mínima com Restrição de Saltos, proposto por Gouveia (1996), que é um Problema de Otimização Combinatória NP-Difícil. Por outro lado, se o número máximo de saltos for zero, isto é, $H = 0$, o problema se reduz ao Problema do Caixeiro Viajante (Dantzig *et al.*, 1954), que também é NP-Difícil.

Antes de apresentar uma formulação de Programação Linear Inteira Mista para o PARST, é descrito o modelo de energia usado nesse trabalho.

3.2 Modelo de consumo de Energia

Para esse trabalho, o consumo de energia do nó sensor foi modelado de forma simplificada para a avaliação experimental de nossa proposta, mas considera a distância e a quantidade de dados transmitidos e recebidos pelos sensores. Park *et al.* (2001) propõem três modelos baseados no comportamento de descarga da bateria: modelo linear, modelo dependente da taxa de descarga e modelo relaxado. Nesse trabalho é utilizado o modelo de descarregamento linear, onde a diferença de potencial é constante ao longo do tempo de vida da bateria. A métrica utilizada para indicar a capacidade da bateria é dada em Ah (Ampere $\times$ hora). Considerando uma bateria com capacidade C_b em Ah, e um sistema com corrente de descarga I em A, o cálculo do tempo de vida teórico da bateria T pode ser medido

pela equação:

$$T = \frac{C_b}{I} \tag{3.1}$$

No modelo linear a bateria é tratada como um armazenador linear de corrente (Park *et al.*, 2001) e sua capacidade C_b depois de uma operação com duração de tempo t_d pode ser calculada pela equação:

$$C_b = C_b' - \int_{t_0}^{t_0+t_d} I(t)dt \tag{3.2}$$

onde C_b' é a capacidade inicial da bateria em Ah e $I(t)$ é a corrente consumida pelo circuito em A por t horas. O modelo linear assume que a corrente $I(t)$ é constante. Neste caso a Equação 3.2 torna-se:

$$C_b = C_b' - \int_{t_0}^{t_0+t_d} I(t)dt = C_b' - It \mid_{t_0}^{t_0+t_d} = C_b' - It_d \tag{3.3}$$

Em redes sem fio, a propagação das ondas eletromagnéticas pode ser modelada em função da distância entre o transmissor e o receptor. Além disso, se não houver nenhum caminho direto (espaço livre) entre o transmissor e receptor, as ondas eletromagnéticas terão que desviar dos obstáculos (construções, montanhas, árvores, etc.) presentes no ambiente. Devido às múltiplas reflexões dos obstáculos, as ondas eletromagnéticas percorrerão diferentes caminhos com tamanhos variados, chegando ao destino em momentos diferentes. Este fenômeno é conhecido como desvanecimento (do inglês *fading*) multicaminho (Heinzelman *et al.*, 2002). E, como dito por Rappaport (2001), independente do modelo usado (espaço livre ou multicaminho), a potência recebida diminui à medida que a distância entre o transmissor e o receptor aumenta. Nos experimento realizados nessa dissertação foram usados tanto o modelo de espaço livre quanto o modelo de multicaminho, dependendo da distância entre os nós sensores que estão se comunicando.

Como mencionado anteriormente, neste trabalho a energia gasta pelo sensor é proporcional à quantidade de dados transmitidos e recebidos, o que é típico para os modelos de otimização matemática propostos para o roteamento de mensagens em RSSF (Keskin *et al.*, 2011). Assume-se também que os nós sensores estão ativos (acordados) somente enquanto estão transmitindo ou recebendo informações sensoriadas e permanecem em modo de espera caso contrário (neste caso, não considera o consumo de energia). Uma mensagem enviada por um nó sensor é recebida por todos os nós sensores que estiverem no raio de comunicação do sensor transmissor. Entretanto, considera-se a energia gasta apenas para os dados realmente destinados ao nó sensor, ou seja, desconsidera-se a energia gasta para ouvir as informações enviadas de outros nós sensores. O sensor 1 é considerado como estação base, como mostrado na Figure 2.8, que se supõem ter energia ilimitada.

Normalmente, a métrica de energia é composta de duas partes: energia gasta para transmitir dados (Et) e energia gasta para receber dados (Er), como é ilustrado pela Figura 3.1. O custo da comunicação entre dois nós sensores geralmente depende da dis-

tância d entre eles, do tamanho da mensagem k, e de um fator f que varia de acordo com as características do ambiente. Este fator é associado à presença de obstáculos na região de sensoriamento e à distância de transmissão (Heinzelman *et al.*, 2002). Se a distância entre os sensores que estão se comunicando for menor que uma determinada distância ($d_{crossover}$), o modelo de espaço livre é usado, neste caso $f = 2$ (atenuação d^2). Caso contrário, o modelo de multicaminho é usado, e $f = 4$ (atenuação d^4).

Cada sensor (exceto a estação base) coleta k bits do campo de sensoriamento e envia seus dados diretamente para outro sensor, que pode ser: um nó sensor comum da rede, um *cluster head*, ou para o agente móvel, que é o caso dos *cluster heads*. Assim, se um sensor i recebe Q_i pacotes de outro sensor, ele envia $Q_i + 1$ pacotes, uma vez que deve adicionar os seus próprios dados, antes de passar adiante os dados recebidos. Então, o custo total estimado E_c de energia consumida, baseado em Heinzelman *et al.* (2002), pode ser calculado com as seguintes equações:

$$Et_i = \begin{cases} k(Q_i + 1)(E_{elec} + \epsilon_{fs} d_{ij}^2) & \text{se } d_{ij} < d_{crossover} \\ k(Q_i + 1)(E_{elec} + \epsilon_{mp} d_{ij}^4) & \text{caso contrário} \end{cases} \tag{3.4}$$

$$Er_i = Q_i k E_{elec} \tag{3.5}$$

$$E_c = \sum_{i \in \mathcal{S} \setminus \{s_1\}} (Et_i + Er_i) \tag{3.6}$$

onde:

- Et_i: Energia gasta pelo sensor i para transmitir $Q_i + 1$ pacotes de k bits ao sensor j a uma distancia d_{ij};
- Er_i: Energia consumida pelo nó sensor i para receber Q_i pacotes de k bits;
- E_{elec}: Constante relacionada à eletrônica do rádio para transmitir ou receber dados;
- ϵ_{fs} e ϵ_{mp}: Constantes associadas à atenuação do sinal, para o modelo de espaço livre e o modelo de multicaminho, respectivamente;
- $\mathcal{S}$: Conjunto de sensores, onde s_1 é a estação base;
- $d_{crossover}$: Distância que determina o modelo de atenuação de sinal a ser utilizado.
- d_{ij}: Distância Euclidiana entre os nós sensores i e j;
- Q_i: Número de pacotes recebidos pelo sensor i;
- E_c: Energia total consumida pela RSSF.

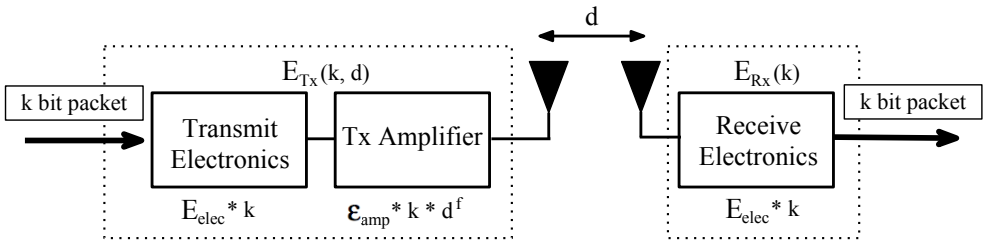


Figura 3.1: Modelo de dissipação de energia proposto em Heinzelman *et al.* (2002)

Observe que a energia total gasta não inclui a energia gasta pela estação base, uma vez que ela não possui restrição de energia, ou seja, tecnicamente sua energia é ilimitada.

Definido o modelo de energia adotado no trabalho, agora pode-se apresentar a formulação de Programação Linear Inteira Mista para o problema.

3.3 Uma Formulação Matemática para o PARST

Nesse trabalho é proposta uma formulação de Programação Linear Inteira Mista, baseada da formulação de Bechelane *et al.* (2009), na qual os dados sensoriados pelos nós sensores são enviados através de caminhos com menor consumo de energia dos sensores para o agente móvel. Como dito anteriormente, este modelo deve limitar o número de saltos, ou seja, os dados são enviados através do caminho se o sensor situa-se perto do *cluster head* (raiz da árvore), ao qual o sensor esta ligado, de tal modo que o percurso do sensor para o *cluster head* obedece a condição de limite de salto.

Com o objetivo de modelar a rede, considera-se um grafo $D = (V, A)$, onde os vértices em $V = \{1, 2, \ldots, n\}$ representam os nós sensores, e A o conjunto completo de arestas $(i,j) \in A$. Para formular o problema como um PLIM, seja $\overline{D} = (\overline{V}, \overline{A})$ um grafo obtido de $D = (V, A)$ adicionando: (i) um vértice artificial 0 e (ii) arestas artificiais $\{\{i, 0\} : i \in V\}$, com $d_{i0} = H_{AM}$ resultando em $\overline{V} = V \cup 0$ e $\overline{A} = A \cup \{\{i,0\} : i \in V, d_{i0} = H_{AM}\}$.

Note que uma floresta restrita a H saltos em D pode ser representada por uma árvore restrita a *H+1* em $\overline{D}$, enraizada em 0. Por esse motivo nossa formulação consiste em uma árvore restrita a $H + 1$ saltos enraizada em 0.

O grafo $\overline{D}$, junto com $Et_i, i \in V$ e $Er_i, i \in V$ definem a *rede de energia*. Já o grafo $D = (V, A)$, junto com as distâncias $\{d_{ij} : i, j \in V\}$, define a *rede de translação*. A Figura 3.2 ilustra a topologia proposta considerando $H = 3$, onde as linhas pontilhadas representam a rede de energia e as linhas contínuas, a rede de translação. As setas indicam o sentido do fluxo das informações pelas duas redes. No caso da rede de energia, esse sentido é o contrário ao da transmissão das informações sensoriadas na RSSF. Observe que apenas os nós raízes das árvores são visitados pelo agente móvel, e também são os únicos que possuem conexão direta com o nó artificial 0.

Para modelar o PARST como um PLIM, são considerados os seguintes dados. Vários

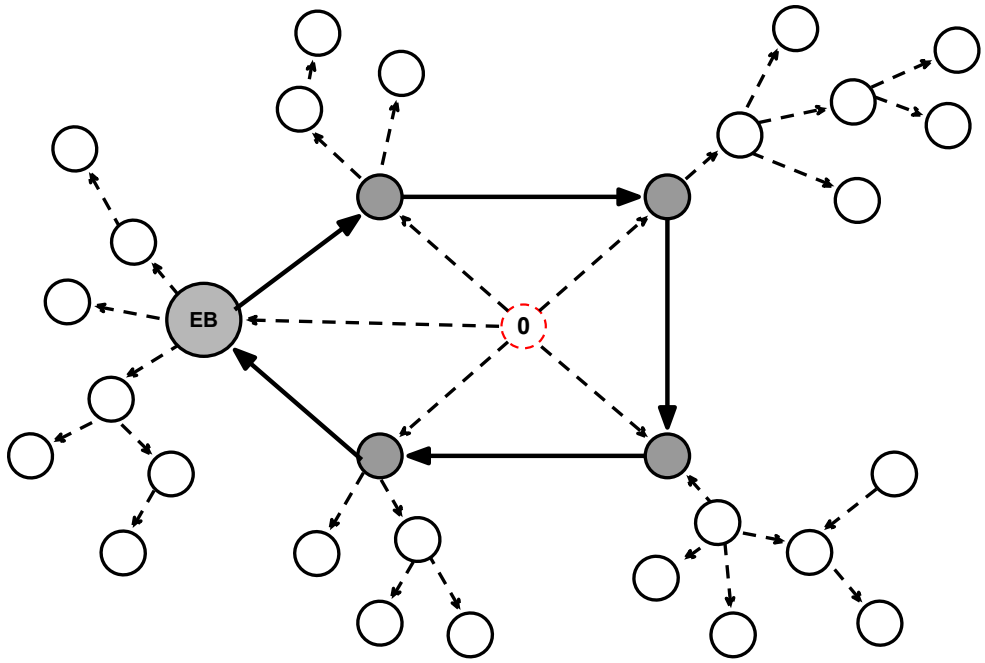


Figura 3.2: Exemplo da arquitetura proposta para o PARST com $H = 3$, adaptado de Bechelane (2009).

foram definidos anteriormente, mas são repetidos por comodidade:

- n: Número de sensores;
- k: Tamanho de cada pacote de dados sensoriado por cada nó sensor, em bits;
- v_{am}: Velocidade constante do agente móvel, em m/s;
- d_{ij}: Distância Euclidiana entre os sensores i e j, dada em metros;
- T^{ch}: Tempo de permanência do agente móvel em cada *cluster head*, em segundos;
- R^c: Limite de distância para que dois sensores possam se comunicar, em metros;
- T_{max}: Tempo (segundos) máximo que o agente móvel pode gastar na rota;
- H: Número máximo de saltos permitido;
- H_{AM}: Distância (metros) entre cada *cluster head* e o agente móvel para ocorrer a transmissão de dados entre eles ($0 \leq H_{AM} \leq R^c$).

Seja $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ o conjunto de nós sensores e $\mathcal{S}' = \mathcal{S} \cup \{0\}$, onde 0 é o nó sensor artificial. A formulação proposta em PLIM usa as seguinte variáveis:

- y_i: variável binária, indicando se o nó sensor $i \in \mathcal{S}$ é ou não um *cluster head*;
- w_{ij}: variável binária, indicando se o *cluster head* j é visitado logo após o *cluster head* i ou não;

- z_{ij}: variável binária, indicando se o arco $(i,j) \in \overline{A}$ pertence a rede de energia restrita a $H+1$ saltos ou não;
- x_{ij}^s: fluxo no arco $(i,j) \in A$ na rede de translação. Considera-se uma unidade de fluxo para cada *cluster head* $s \in \mathcal{S}\backslash\{s_1\}$, que deve fluir do sorvedouro até o sensor;
- f_{ij}^s: fluxo de $s \in \mathcal{S}$ no arco $(i,j) \in \overline{A}$ na rede de translação. Considera-se uma unidade de fluxo s fluindo do nó 0 até o sensor s;
- Q_i: Número de pacotes recebidos pelo nó sensor $i \in \mathcal{S}$.

Como energia consumida estimada, a formulação usa as equações (3.4)-(3.6), mas a variável z_{ij} é adicionada em (3.4) resultando em:

$$Et_i = \begin{cases} k(Q_i+1)(E_{elec} + \epsilon_{fs} \sum_{j \in \mathcal{S}'} z_{ij} d_{ij}^2) & \text{se } d_{ij} < d_{crossover} \\ k(Q_i+1)(E_{elec} + \epsilon_{mp} \sum_{j \in \mathcal{S}'} z_{ij} d_{ij}^4) & \text{caso contrário} \end{cases} \quad \forall i \in \mathcal{S}\backslash\{s_1\} \tag{3.7}$$

Dessa forma, a formulação matemática completa é dada pelas equações (3.5)-(3.7), a função objetivo (3.8) e o conjunto de restrições (3.9)-(3.31), que são explicados a seguir:

$$\min E_c = \sum_{i \in \mathcal{S}\backslash\{s_1\}} (Et_i + Er_i) \tag{3.8}$$

$$\sum_{i \in \mathcal{S}\backslash\{s_1\}} x_{1i}^s = y_s, \qquad \forall s \in \mathcal{S}\backslash\{s_1\} \tag{3.9}$$

$$\sum_{j \in \mathcal{S}} x_{ij}^i - \sum_{j \in \mathcal{S}} x_{ji}^i = -y_i, \qquad \forall i \in \mathcal{S}\backslash\{s_1\} \tag{3.10}$$

$$\sum_{j \in \mathcal{S}} x_{ij}^v - \sum_{j \in \mathcal{S}} x_{ji}^v = 0, \qquad \forall i,v \in \mathcal{S}\backslash\{s_1\}, i \neq v \tag{3.11}$$

$$x_{ij}^s \leq w_{ij}, \qquad \forall i,j \in \mathcal{S}, s \in \mathcal{S}\backslash\{s_1\} \tag{3.12}$$

$$w_{ij} \leq y_i, \qquad \forall i,j \in \mathcal{S} \tag{3.13}$$

$$w_{ij} \leq y_j, \qquad \forall i,j \in \mathcal{S} \tag{3.14}$$

$$\sum_{i,j \in \mathcal{S}} \frac{w_{ij} d_{ij}}{v_{am}} + T^{ch} \sum_{i \in \mathcal{S}\backslash\{s_1\}} y_i \leq T_{max} \tag{3.15}$$

$$\sum_{j \in \mathcal{S}\backslash\{s_1\}} w_{1j} \leq 1 \tag{3.16}$$

$$\sum_{j \in \mathcal{S}\backslash\{s_1\}} w_{1j} = \sum_{j \in \mathcal{S}\backslash\{s_1\}} w_{j1} \tag{3.17}$$

$$y_1 = 1 \tag{3.18}$$

$$\sum_{j \in \mathcal{S}, j \neq i} w_{ij} \leq y_i \qquad \forall i \in \mathcal{S}\backslash\{s_1\} \tag{3.19}$$

$$\sum_{j\in\mathcal{S},j\neq i} w_{ji} = y_i \qquad \forall i \in \mathcal{S}\backslash\{s_1\} \tag{3.20}$$

$$\sum_{i,j\in\mathcal{S}'} z_{ij} = n \tag{3.21}$$

$$\sum_{j\in\mathcal{S}} f_{0j}^q = 1 \qquad \forall q \in \mathcal{S} \tag{3.22}$$

$$\sum_{j\in\mathcal{S}'} f_{ij}^q - \sum_{j\in\mathcal{S}'} f_{ji}^q = 0 \qquad \forall i,q \in \mathcal{S}, i \neq q \tag{3.23}$$

$$\sum_{j\in\mathcal{S}'} f_{ij}^i - \sum_{j\in\mathcal{S}'} f_{ji}^i = -1 \qquad \forall i \in \mathcal{S} \tag{3.24}$$

$$f_{ij}^q \leq z_{ij} \qquad \forall q \in \mathcal{S}, \forall i,j \in \mathcal{S}' \tag{3.25}$$

$$\sum_{i,j\in\mathcal{S}'} f_{ij}^q \leq H + 1 \qquad \forall q \in \mathcal{S} \tag{3.26}$$

$$\sum_{i\in\mathcal{S}} z_{ij} + y_j = 1 \qquad \forall j \in \mathcal{S} \tag{3.27}$$

$$z_{0i} = y_i \qquad \forall i \in \mathcal{S} \tag{3.28}$$

$$z_{ij} d_{ij} \leq R^c \qquad \forall i,j \in \mathcal{S} \tag{3.29}$$

$$Q_s = \sum_{i,j\in\mathcal{S}} f_{si}^j \qquad \forall s \in \mathcal{S} \tag{3.30}$$

$$\begin{aligned}
&x_{ij}^s \in \mathbb{R}^+, && \forall (i,j) \in A, s \in \mathcal{S}\backslash\{s_1\} \\
&z_{ij} \in \{0,1\}, && \forall (i,j) \in \overline{A} \\
&y_i \in \{0,1\}, && \forall i \in \mathcal{S} \\
&f_{ij}^q \in \mathbb{R}^+, && \forall (i,j) \in \overline{A}, \forall q \in \mathcal{S} \\
&w_{ij} \in \{0,1\}, && \forall i,j \in \mathcal{S} \\
&Q_i \in \mathbb{N}, && \forall i \in \mathcal{S}
\end{aligned} \tag{3.31}$$

Na formulação acima, o problema de definir a rota do agente móvel (rede de translação) é representado pelas restrições (3.9)-(3.20). Tais restrições representam o Problema de Fluxo de Custo Mínimo (do inglês *Minimum Cost Flow Problem*) definido sobre a rede $D = (V, A)$, o parâmetro T_{max} e as distâncias $\{d_{ij} : i, j \in V\}$. No PARST, cada *cluster head* s tem demanda de uma unidade de fluxo s distinto. Assim, cada fluxo na rede de translação deve fluir do ponto inicial da rota (nó sensor 1) até seu destino (*cluster head* correspondente) usando um caminho simples.

O conjunto de restrições (3.9) garantem que o fluxo em cada *cluster head* s tenha inicio no vértice 1. Se s não for *cluster head* ($y_s = 0$), não haverá oferta de fluxo em 1 e, assim, não haverá fluxo em s pela rede de translação. As restrições (3.10) e (3.11), por sua vez, garantem a conservação do fluxo em s. O conjunto de restrições (3.12)-(3.14) garantem que a trajetória do agente móvel passe por cada *cluster head*. Enquanto a restrição (3.15) limita

superiormente o tempo total da rota do agente móvel. As restrições (3.16)-(3.18) definem que a rota, caso ela exista, começa e termina no vértice 1 (a estação base), garantindo que a rota é uma ciclo hamiltoniano. Finalmente, as restrições 3.19) e (3.20) evitam que o agente móvel passe por nó sensores que não são *cluster heads*.

A árvore restrita a *H+1* saltos (rede de energia) é definia pelas restrições (3.21)-(3.29), baseada na formulação apresentada por Gouveia (1996). Também é empregada uma unidade de fluxo para cada vértice de $\mathcal{S}$ disponibilizada no vértice artificial 0. A demanda de cada uma delas é distribuída em cada um dos nós sensores. Neste caso, cada unidade de fluxo na rede de energia deve fluir da origem (vértice artificial 0) ao seu destino (sensor correspondente) usando um caminho que não emprega mais de $H+1$ arcos.

A equação (3.21) define o número total de arcos usados na rede de energia. As restrições (3.22) garantem que a árvore restrita a $H+1$ saltos seja enraizada no nó artificial 0. O conjunto de restrições (3.23) e (3.24) garantem a conservação de fluxo pela árvore. As inequações (3.25) obrigam que um arco $[i,j]$ só pode ser utilizado se ele fizer parte da árvore restrita a H saltos, enquanto as restrições (3.26) garantem que não haja mais do que $H+1$ saltos entre qualquer $i \in \mathcal{S}$ e o vértice 0 na árvore de comunicação. Pelas restrições (3.27) e (3.28), nenhuma aresta pode ser incidente ao *cluster head*, exceto arestas vindos do nó artificial 0. O arco $[i,j] \in A$ deve existir apenas se a distância entre os nós sensores i e j for menor ou igual ao raio de comunicação, isto é garantido pelas inequações (3.29). Finalmente, as restrições (3.30) definem o número de pacotes recebidos pelo sensor $s \in \mathcal{S}$ e o conjunto de restrições (3.31) definem os domínios das variáveis de decisão.

Entretanto, esta formulação é não-linear, pois as variáveis binárias z_{ij} são multiplicadas pelas variáveis contínuas Q_i em (3.7). Para evitar a não-linearidade resultante, são definidas as variáveis δ_{ij}, onde $\delta_{ij} = (Q_i+1)z_{ij}$, e as restrições (3.7) são substituídas pelas seguintes[1]:

$$Et_i = \begin{cases} k((Q_i+1)E_{elec} + \epsilon_{fs}\sum_{j\in\mathcal{S}'}\delta_{ij}d_{ij}^2), & \text{se } d_{ij} < d_{crossover} \\ k((Q_i+1)E_{elec} + \epsilon_{mp}\sum_{j\in\mathcal{S}'}\delta_{ij}d_{ij}^4), & \text{caso contrário} \end{cases} \quad \forall i \in \mathcal{S}\backslash\{s_1\} \tag{3.32}$$

$$\delta_{ij} \leq n z_{ji} \qquad \forall i \in \mathcal{S}\backslash\{s_1\}, j \in \mathcal{S}' \tag{3.33}$$

$$\delta_{ij} \leq (Q_i+1) \qquad \forall i \in \mathcal{S}\backslash\{s_1\}, j \in \mathcal{S}' \tag{3.34}$$

$$\delta_{ij} \geq (Q_i+1) - n(1-z_{ji}) \qquad \forall i \in \mathcal{S}\backslash\{s_1\}, j \in \mathcal{S}' \tag{3.35}$$

$$\delta_{ij} \in \mathbb{N}, \qquad \forall i \in \mathcal{S}\backslash\{s_1\}, j \in \mathcal{S}' \tag{3.36}$$

Depois de substituir as restrições (3.7) pelo conjunto de restrições (3.32)-(3.36) no modelo, a formulação do problema se torna uma formulação de Programação Linear Inteira

[1]Por questão de simplicidade na notação da nova restrição, $\delta_{ij} = (Q_i+1)z_{ij}$ e não $Q_i z_{ij}$.

Mista. Na próxima seção é apresentada uma abordagem para aumentar o tempo de vida da rede usando esta formulação PLIM para o problema.

3.4 Aumentando o tempo de vida da RSSF

Como visto na Seção 2.1, o tempo de vida da RSSF pode ser medido de vária formas, sendo um deles definido como o tempo até o primeiro sensor na rede morrer. Duas formas muito utilizadas na literatura para medir o tempo de vida são: (i) unidade de tempo: quanto tempo, em segundos por exemplo, a rede ficará em funcionamento até o primeiro sensor morrer, e (ii) número de *rounds* (rodadas) que a rede poderá operar até que o primeiro nó sensor fique sem energia. Nesse trabalho, é usada a ideia de *round* da rede, onde o número de *rounds* representa o número de vezes que o agente móvel pode percorrer a rota por completo até que o primeiro nó sensor esgote sua energia.

A solução encontrada para o PARST, usando a formulação apresentada na Seção 3.3, representa uma configuração para a RSSF (a melhor solução para rede de energia e rede de translação) com o menor consumo total de energia. Dessa forma, para descobrir o número de *rounds* de uma determinada solução para a rede, divide-se a energia inicial dos nós sensores E_{ini} pela energia gasta pelo nó sensor com o maior consumo de energia na solução e_m ($e_m \geq Et_i + Er_i, \forall i \in \mathcal{S} \backslash \{s_1\}$). A formulação exata apresentada para o PARST minimiza o consumo total de energia, mas não maximiza o tempo de vida, pois será definida pelo nó que consome mais energia. O ideal seria balancear o consumo de energia entre os nós minimizando o consumo do nó que consome mais energia. Para isso, uma solução é modelar a formulação matemática minimizando e_m ao invés de minimizar a energia gasta total. Entretanto, esta abordagem normalmente é mais difícil de se resolver, consumindo mais tempo. Outra alternativa é gerar várias soluções (configurações) diferentes, usando a formulação matemática proposta, e escolher aquelas que maximizam o número de *rounds* de forma que a soma da energia gasta por cada nó sensor em cada configuração escolhida seja não superior a energia inicial E_{ini}.

Se todas as possíveis configurações para a RSSF são conhecidas, o problema consiste em definir a quantidade de *rounds* que cada configuração ficará ativa, respeitando a energia disponível dos nós sensores. Entretanto, nem sempre o conjunto completo de configurações para a RSSF é conhecido a priori, e deve, portanto, ser gerado. Geralmente, esse conjunto é bastante grande e, portanto, um tempo considerável seria gasto na sua geração. Além disso, caso se tenha todas as configurações geradas, pode ser difícil resolver o problema de seleção das configurações em um tempo viável, devido ao grande número de configurações possíveis, o que implicaria em um conjunto com muitas variáveis. Nesses casos, é mais aconselhável o uso da técnica de geração de colunas, que gera configurações enquanto resolve o problema de seleção das configurações (Santos, 2008). A grande vantagem da geração de colunas é que a solução ótima pode ser obtida sem se gerar explicitamente todo

o conjunto de configurações. Nesta técnica, o problema é decomposto em dois problemas:

- Problema mestre: define a quantidade de *rounds* que cada configuração ficará ativa, respeitando restrição da quantidade de energia disponível dos nós sensores, porém com um conjunto restrito de colunas (neste caso algumas configurações para a RSSF);
- Subproblema: gerador de novas colunas para o problema mestre (nesse caso, gerador de novas configurações para a RSSF), colunas estas que, quando incorporadas ao conjunto de colunas utilizadas pelo problema mestre, melhoram a solução atual.

3.4.1 Problema Mestre

O problema se resume em determinar o número de *rounds* que cada configuração deve permanecer ativa. Cada coluna da matriz de energia, ou seja, cada variável do problema é uma configuração viável para a RSSF, em que cada nó sensor gasta uma certa quantidade de energia. Existe uma restrição para cada nó sensor, representada por uma linha da matriz. Este problema mestre é modelado da seguinte forma:

$$PM_{RSSF} := max \sum_{i \in \mathcal{C}} \lambda_i \tag{3.37}$$

$$\sum_{i \in \mathcal{C}} a_{ij}\lambda_i \leq E_{ini}, \qquad \forall j \in \mathcal{S}\backslash\{s_1\} \tag{3.38}$$

$$\lambda_i \in \mathbb{N}, \qquad \forall i \in \mathcal{C} \tag{3.39}$$

Neste modelo, $\mathcal{C}$ é o conjunto de configurações viáveis e $\mathcal{S}$ é o conjunto de nós sensores da rede. Para cada $i \in \mathcal{C}$ é associada uma variável de decisão inteira λ_i, cujo valor indica o número de *rounds* que esta configuração ficará ativa na rede. A matriz real A, de dimensões $|\mathcal{C}| \times |\mathcal{S}\backslash\{s_1\}|$, com valor a_{ij} representando o gasto energético do sensor j na configuração i. A função objetivo (3.37) procura maximizar o número total de *rounds* para a RSSF, de forma que a soma da energia gasta por cada sensor nas configurações selecionadas não seja superior a energia inicial, o que é garantido pelas restrições (3.38). E por fim, (3.39) estabelece o tipo das variáveis de decisão.

Para obter os preços duais, necessários na solução do subproblema, resolve-se a relaxação linear de (3.37)-(3.39), visto que este é um problema de programação linear inteira, não tendo um problema dual associado a ele cuja solução tenha, garantidamente, o mesmo valor. Nesse caso, as variáveis correspondentes a cada coluna deixam de ser inteiras e passam a ser não-negativas. Como este é o problema mestre num método de geração de colunas, em vez do conjunto completo $\mathcal{C}$, o modelo usa um subconjunto de configurações $\overline{\mathcal{C}} \subseteq \mathcal{C}$. Assim, o problema mestre relaxado (PMR) é modelado da seguinte forma:

$$PMR_{RSSF} := max \sum_{i \in \overline{\mathcal{C}}} \lambda_i \tag{3.40}$$

$$\sum_{i \in \overline{\mathcal{C}}} a_{ij}\lambda_i \leq E_{ini}, \qquad \forall j \in \mathcal{S}\backslash\{s_1\} \tag{3.41}$$

$$\lambda_i \in \mathbb{R}, \qquad \forall i \in \overline{\mathcal{C}} \tag{3.42}$$

3.4.2 Subproblema

O objetivo do subproblema é gerar uma (ou mais) nova configuração para a RSSF, que seja viável e de custo reduzido positivo, quando considerados os preços duais das restrições associadas aos nós sensores fornecidos pelo problema mestre anteriormente resolvido. Estas novas configurações são novas colunas acrescentadas ao problema mestre, que melhoram o valor da solução, ou atualizam os preços duais para permitir a geração de outras novas configuração pelo subproblema.

Resolvendo o PMR_{RSSF} obtêm-se um conjunto de variáveis duais π_j, $\forall j \in \mathcal{S}\backslash\{s_1\}$ associadas às restrições de energia (3.41), para a solução do problema com um conjunto $\overline{\mathcal{C}}$ de configurações consideradas. Usando os valores das variáveis duais, pode-se calcular o custo reduzido máximo. Para encontrar uma coluna (configuração), tendo o custo reduzido máximo, substitui-se a função objetivo (3.8) na formulação PARST pela seguinte, que inclui os preços duais:

$$SUB_{RSSF} := max \left(1 + \sum_{j \in \mathcal{S}\backslash\{s_1\}} (Et_j + Er_j)\pi_j\right) \tag{3.43}$$

Sujeito a

(3.9)-(3.36)

Para cada execução do subproblema tem-se o consumo de energia de cada nó sensor $(Et_j + Er_j)$ $\forall j \in \mathcal{S}\backslash\{s_1\}$, que é associado a a_{ij} $\forall j \in \mathcal{S}\backslash\{s_1\}$, onde i representa o número da configuração atual. Se a solução gerada pelo SUB_{RSSF} possuir custo positivo, a configuração (coluna) correspondente é adicionada ao problema mestre, que é novamente resolvido, e têm-se novos preços duais para buscar outra configuração, com os custos dos nós sensores alterados. Quando a solução tiver custo zero ou negativo, então a solução do problema mestre é a solução ótima do problema relaxado, como mostrado pela Figura 3.3.

3.4.3 Branch-and-Price

O método de geração de colunas resolve o PMR_{RSSF}, ou seja, a relaxação linear do problema PM_{RSSF}. Quando a solução encontrada não é inteira, parte-se para uma enumeração

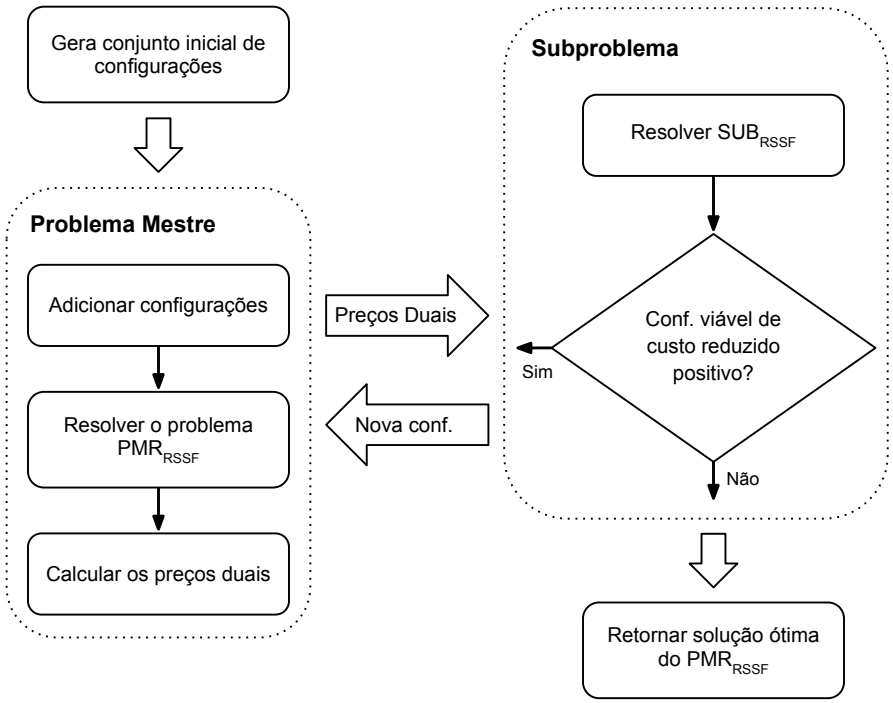


Figura 3.3: Interação dos problemas mestre e subproblema no algoritmo de geração de colunas, adaptado de Santos (2008).

da árvore de *branch-and-bound*. Se a geração de colunas (*princing*) continua a ser usada em cada nó da árvore, o método é chamado de *branch-and-price*. Neste trabalho opta-se por não fazer o *branch-and-price* devido a seu elevado custo computacional, aumentando mais ainda o tempo total gasto na geração de colunas. Ao final da geração de colunas, o problema PM_{RSSF} é resolvido considerando apenas as colunas já adicionadas, desta vez com as variáveis λ_i novamente inteiras.

3.5 Comentários Finais

Nesse capítulo foram apresentadas uma formulação em grafos, uma formulação de Programação Linear Inteira Mista e uma abordagem para aumentar o número de *rounds* da rede para o PARST.

Para resolver o PARST de forma exata foi utilizado o pacote de otimização comercial ILOG CPLEX Optimization (2011). Entretanto, o tempo computacional para resolver o problema é elevado, mesmo para instâncias com poucos nós sensores. Nesse sentido, no próximo capítulo são apresentadas duas metaheurísticas para auxiliar a resolução do PARST.

Capítulo 4

Método híbrido para o PARST

Como visto no capítulo anterior, a solução para o PARST é formada pela rede de energia, composta pelos *cluster heads* e árvores de comunicação entre os nós sensores, e a rede de translação, composta pela rota do agente móvel. Assim, uma boa solução para o PARST é composta por três importantes aspectos:

- um bom conjunto de *cluster heads*;
- uma boa floresta de comunicação;
- uma boa rota para o agente móvel;

Os dois primeiros estão relacionados à minimização da energia consumida e o último (junto com o primeiro) está relacionado com o atraso na entrega dos dados. É fácil perceber que todos os três estão relacionados um com os outros. A rota do agente móvel visita os *cluster heads* e a floresta de comunicação é construída usando estes *cluster heads* como raízes de cada sub-árvore. Dessa forma, para se definir a rota e a floresta, é preciso saber quais nós sensores são selecionados para serem *cluster heads*. Por outro lado, os *cluster heads* devem ser cuidadosamente selecionados, de forma a se reduzir o consumo de energia e ao mesmo tempo não prejudicar a rota do agente móvel pelo atraso na entrega dos dados.

Como esperado, os tempos computacionais gastos pela formulação matemática proposta no capítulo anterior foram elevados, mesmo para redes com poucos nós sensores. Dessa forma, são propostos outros métodos para tratar o PARST: métodos heurísticos para resolver cada subproblema, mas de forma cooperativa. Para os dois últimos objetivos, definição da floresta de comunicação e rota do agente móvel, são usadas heurísticas construtivas gulosas com Busca Local. Para o primeiro, a seleção dos *cluster heads*, são apresentadas duas metaheurísticas: Algoritmo Genético e GRASP. Os métodos propostos trabalham cooperativamente da seguinte forma: cada solução potencial (conjunto de *cluster heads*) gerada, tanto pelo Algoritmo Genético quanto pelo GRASP, deve ser avaliada; o valor da solução depende do consumo de energia e do atraso na entrega dos dados, e estes valores são calculados pelas heurísticas construtivas.

Os métodos propostos para cada subproblema são detalhados nas próximas seções.

4.1 Herística para a floresta de comunicação

O objetivo dessa etapa é construir, se for possível, a árvore de comunicação entre os nós sensores usando no máximo H saltos até os nós sensores que são *cluster heads*. Além de construir, o método construtivo deve também avaliar a árvore gerada em relação a energia gasta total.

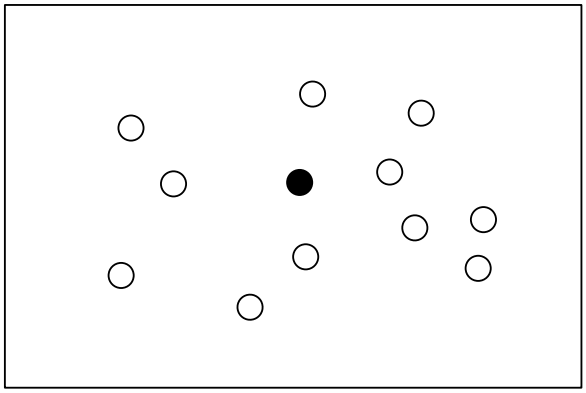
Neste trabalho são propostos dois métodos construtivos para gerar e avaliar a árvore de comunicação, denominados método *H-passos* e método *Prim-H*, explicados a seguir.

4.1.1 Método *H-passos*

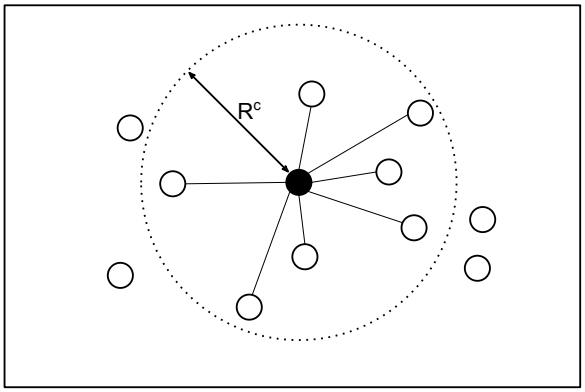
Uma forma bem simples de construir a floresta de comunicação é fazer o seguinte: primeiramente conecta-se todos os nós sensores $i \in \mathcal{S}$, que não são *cluster heads*, ao *cluster head* $j \in \mathcal{S}, i \neq j$ mais próximo, tal que $d_{ij} \leq R^c$. Então, se algum sensor não for capaz de se comunicar com algum *cluster head*, conecta-se este nó sensor ao mais próximo sensor localizado em seu raio de comunicação e esteja conectado (usando um ou mais saltos) a um *cluster head*. Este segundo passo é repetido até que todos os sensores estejam conectados e, assim, possam enviar os seus dados a algum *cluster head* ou até que o número de passos seja menor ou igual a H. Assim, é construída uma floresta de comunicação e pode-se computar a energia gasta por esta rede usando as equações (3.4), (3.5) e (3.6). Caso contrário, a solução é marcada como inválida e recebe uma penalização, que será usada nas etapas das metaheurísticas apresentadas na Seção 4.3. Este método é chamado de *H-passos*.

Para exemplificar o método *H-passos*, considere a rede mostrada na Figura 4.1a e número de saltos máximo $H = 2$. O nó preto é o *cluster head* da rede, e no primeiro passo, todos os sensores que possuem algum *cluster head* em seu raio de alcance são conectados a ele, como mostra a Figura 4.1b. Caso mais de um *cluster head* esteja no raio de alcance de um sensor, este se conecta ao mais próximo. Como alguns sensores ficaram desconectados, o segundo passo é executado: para cada sensor que não está diretamente ligado ao *cluster head* (usando um salto), ele é conectado ao nó sensor mais próximo que já esteja conectado a algum *cluster head*, como ilustrado pela Figura 4.1c. Se $H > 2$, o segundo passo é repetido $H - 1$ vezes ou até que todos os sensores estejam conectados.

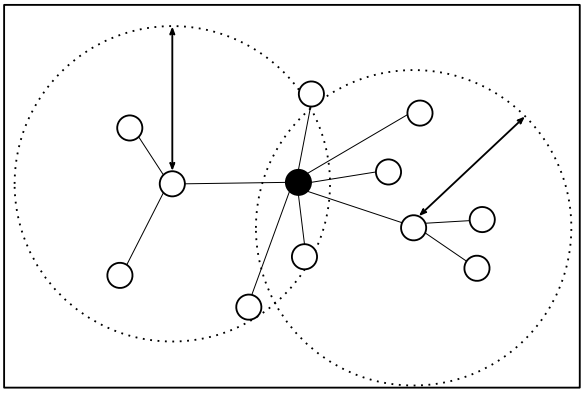
Este método construtivo é bem simples e de fácil implementação, conseguindo encontrar uma rede de comunicação com o menor número de saltos possível. Entretanto, esta abordagem não garante a melhor floresta em relação a energia consumida, por isso uma busca local (detalhada na Seção 4.4.2) é proposta para melhorar a solução encontrada.



(a) Configuração inicial



(b) Primeiro passo



(c) Segundo passo

Figura 4.1: Exemplo do método *H-passos*, com $H = 2$.

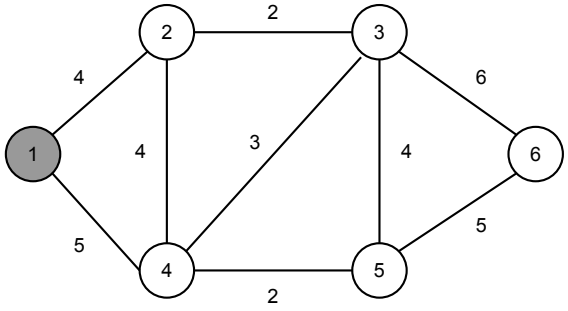
4.1.2 Método *Prim-H*

Este método é baseado na heurística construtiva apresentada por Gouveia (1996), a qual é explicada detalhadamente nesta seção.

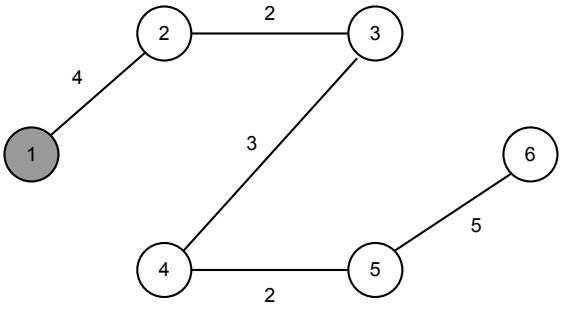
A ideia do método *Prim-H* é transformar o problema da árvore geradora em um problema de árvore geradora com restrição de saltos usando o Algoritmo Prim. Para isso, considere um nó artificial 0, de forma que apenas os *cluster heads* se conectam a esse nó. Assim, uma floresta restrita a H saltos pode ser representada por um árvore restrita a $H + 1$ saltos enraizada em 0. No algoritmo de *Prim-H*, além de escolher a aresta (u, v) de custo mínimo que não forma ciclo (como no Prim comum), também é verificado se o número de saltos, de u até o *cluster head* ao qual ele está conectado, não é maior que $H + 1$. Para acelerar o cálculo do número de saltos, é utilizado um vetor h que armazena o número de saltos de cada nó e este é atualizado toda vez que um nó for conectado a outro que já esteja previamente conectado. Se $h(v) = x$, e u se conecta a v, então $h(u) = x + 1$.

A Figura 4.2 compara o algoritmo de Prim convencional (Figura 4.2b) com o *Prim-H* (Figura 4.2c), onde os pesos nas arestas representam a distância entre os nós sensores. Como pode-se perceber, ao invés do nó 5 se conectar ao nó 4, como em Figura 4.2b, ele se conecta ao nó 3, uma vez que $h(4) + 1 \nleq H + 1$, para $H = 3$. O mesmo acontece com o nó 6, que também não se conecta a outro nó, previamente conectado, que possui a aresta

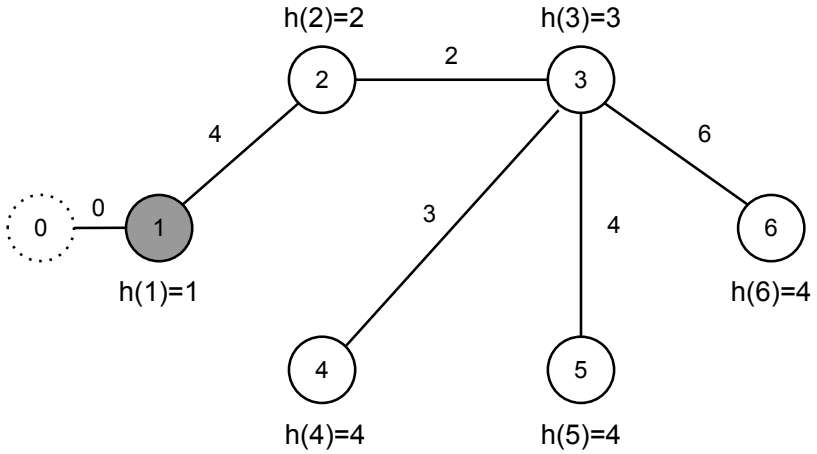
de menor peso devido a restrição de saltos.



(a) Configuração inicial



(b) Algoritmo de Prim



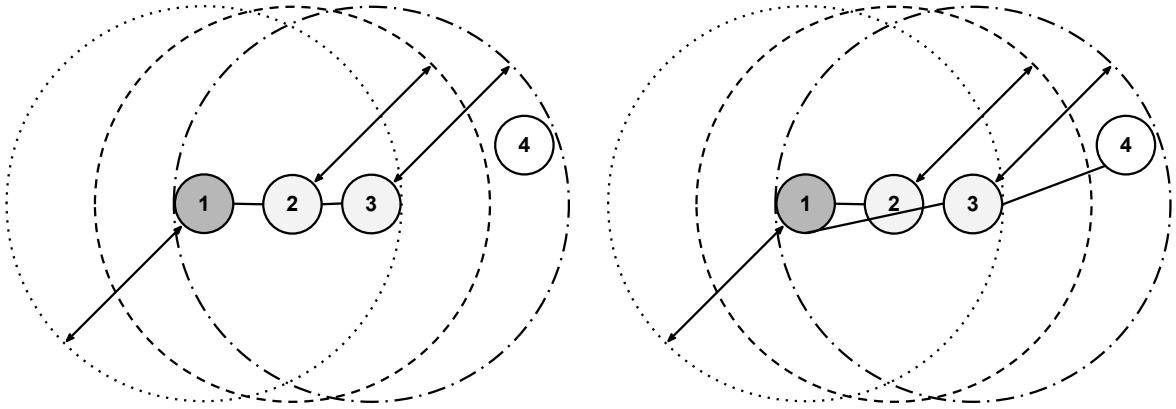
(c) Algoritmo Prim-H

Figura 4.2: Comparação entre o Prim e o Prim-H, considerando $H = 3$. Observe que os nós 5 e 6 não usam as arestas de menor distância por esta conexão violar a restrição de saltos.

Como no método *H-passos*, esta abordagem não garante a floresta ótima em relação a energia consumida. Na Figura 4.2c o nó 4 se conecta ao nó 3, e como a energia gasta para enviar dados é proporcional a distância ao quadrado ou a quarta, a energia gasta para que os dados do nó 4 cheguem ao nó 1 é proporcional a 29. Entretanto, se ele se conectasse ao nó 1, a energia consumida seria proporcional a 25. Dessa forma, como no *H-passos*, também é aplicada uma busca local (detalhada na Seção 4.4.2) na solução encontrada pelo *Prim-H*.

A heurística *Prim-H* geralmente encontra uma rede de comunicação que gasta menos energia que a heurística *H-passos*, entretanto para isso podem ser necessários mais saltos que o *H-passos*. Por esse motivo, o *Prim-H* pode não encontrar uma rede de comunicação com H saltos, mesmo ela existindo, como ilustra a Figura 4.3a considerando $H = 2$. Observe que ao se conectar o sensor 3 ao sensor mais próximo, o sensor 2, ele usará dois saltos. Como o sensor 4 só consegue se comunicar com o sensor 3, que já possui $h(3) = 2$, ele fica desconectado da rede. Este problema não ocorre com o *H-passos*, como mostra

a Figura 4.3b. Para contornar este problema, quando algum nó fica desconectado, este é conectado ao nó sensor (*cluster head* ou sensor comum) mais próximo mesmo que tenha que violar a restrição de saltos, deixando a cargo da busca local tentar corrigir este problema. Neste caso, como no método *H-passos*, a solução é marcada como inválida e recebe uma penalização.



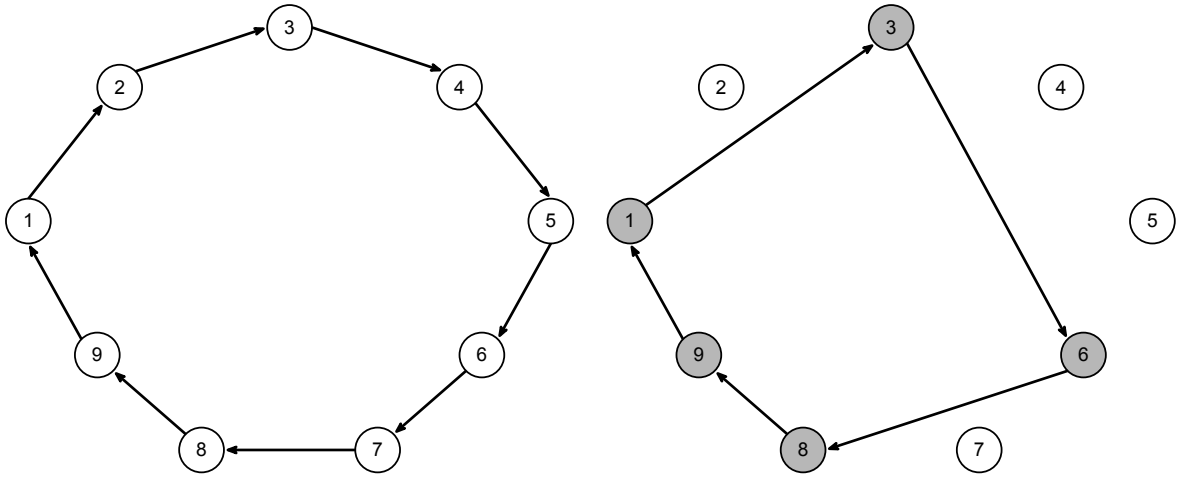
(a) Usando o *Prim-H* (b) Usando o *H-passos*

Figura 4.3: Problema encontrado pelo *Prim-H* para uma RSSF com $H = 2$, deixando a rede desconectada, mesmo havendo uma solução.

4.2 Heurística para a rota do agente móvel

Definidos os *cluster heads*, este método construtivo visa construir a rota do agente móvel com a menor distância total, ou seja, encontrar uma solução para um TSP (Dantzig *et al.*, 1954) considerando apenas os *cluster heads* como nós do conjunto de entrada. Dessa forma, o agente móvel gastará o menor tempo para recolher os dados sensoriados pela RSSF, considerando os *cluster heads* selecionados. Como o número de *cluster heads*, comparado com o número de nós sensores da rede, é normalmente bem menor, esta fase pode ser resolvida de forma exata. Entretanto, ela deve ser calculada para cada solução gerada durante o GRASP ou para cada indivíduo do AG, o que consumiria muito tempo. Com o objetivo de reduzir o tempo de se resolver repetidamente esta parte, antes do método híbrido iniciar, é pré-computada a solução ótima do TSP para todos os nós. Dado um conjunto de *cluster heads*, uma possível solução é o TSP pré-computado sem os sensores comuns (que não são *cluster heads*). A Figura 4.4 exemplifica esse método. A solução do TSP para a RSSF considerando os 9 sensores é $1 \rightarrow 2 \rightarrow \cdots \rightarrow 8 \rightarrow 9 \rightarrow 1$ (Figura 4.4a), e sendo os nós 1, 3, 6, 8 e 9 *cluster heads*, a rota $1 \rightarrow 3 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 1$ é considerada como a trajetória do agente móvel.

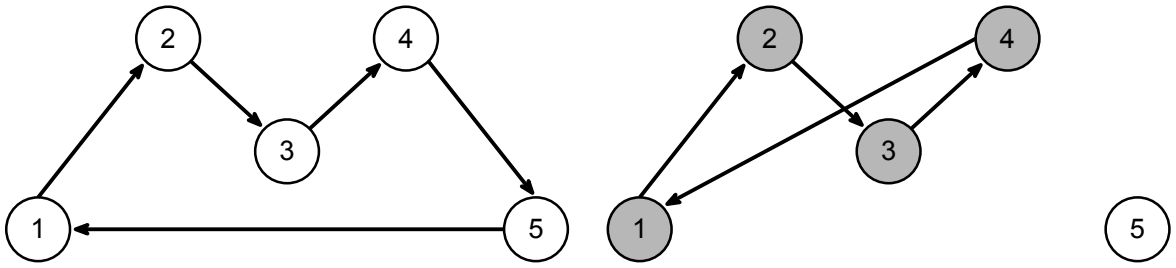
Entretanto, esta abordagem não garante o menor caminho para todos os subconjuntos, como mostrado na Figura 4.5. Assim, uma simples e rápida busca local (detalhada na



(a) TSP considerando todos os nós (b) TSP considerando apenas os CHs

Figura 4.4: Heurística construtiva para construir a rota do agente móvel.

Seção 4.4.3) é aplicada. Se, após a busca local, o tempo gasto pelo agente móvel na rota for maior que T_{max}, a solução é marcada como inviável e penalizada.



(a) TSP considerando todos os nós (b) TSP considerando apenas os CHs

Figura 4.5: Exemplo de mal funcionamento da heurística.

4.3 Heurísticas para a seleção dos *cluster heads*

Como dito anteriormente, os *cluster heads* devem ser cuidadosamente selecionado a fim de se ter uma boa floresta de comunicação e uma boa rota para o agente móvel. Com esse objetivo, são propostas duas heurísticas baseadas em metaheurísticas (Algoritmo Genético e GRASP) que trabalham de forma cooperativa com os métodos construtivos anteriormente apresentados.

4.3.1 Algoritmo Genético

O Algoritmo Genético (AG), proposto por Holland (1975), é um algoritmo de busca heurística adaptativa que simula o processo de seleção genética e eliminação natural na evolução biológica. Em comparação com os algoritmos de busca tradicionais de inteligência artificial

(AI), em geral, um algoritmo genético é capaz de adquirir e acumular automaticamente conhecimento implícito sobre o espaço de busca durante o seu processo de pesquisa. Por conseguir bons resultados e ser de fácil implementação, o AG tem sido frequentemente utilizado para resolver problemas de otimização combinatória.

Um algoritmo genético possui uma população de indivíduos (cromossomos), cada um representando uma possível solução. Como cada solução tem um custo, os cromossomos devem ser avaliados a fim de gerar um valor, tornando possível comparar um com outro qualitativamente. Este valor representa a adaptação (*fitness*) do indivíduo codificado no cromossomo e indivíduos com melhor adaptação são mais propensos a participar nas fases de reprodução e mutação para gerar novos indivíduos com características semelhantes.

Como dito no início do capítulo, uma solução para o problema consiste em: (i) um conjunto de *cluster heads*; (ii) uma floresta de comunicação restrita a H saltos e; (iii) uma rota para o agente móvel. Este último é a solução de um TSP considerando apenas os *cluster heads*. Como o número de *cluster heads* é normalmente pequeno, a rota para cada indivíduo não é codificada nem guardada pelos operadores do AG, sendo determinada pela heurística apresentada na seção anterior, que praticamente não consome tempo. Já para (i) e (ii), uma possível representação seria um vetor de inteiros $[a_1, a_2, \ldots, a_n]$, como ilustrado na Figura 4.6: se $a_i = 0$, então i é um *cluster head*; caso contrário, o sensor i deve enviar seus dados para o sensor a_i. No exemplo dado, o sensor 2 é um *cluster head* e o sensor 3 envia seus dados para o sensor 5, que por sua vez envia seus dados para o *cluster head* 2, como mostra a Figura 4.7. O problema com esta representação é o número potencial de soluções inviáveis. Cada a_i deve ser 0 ou um valor tal que $d_{ia_i} < R^c$, ou seja, ou i é um *cluster head* ou deve enviar seus dados para um sensor que esta localizado dentro do seu raio de comunicação. Além disso, a restrição de H saltos deve ser satisfeita, o que significa que, para cada sensor i, o tamanho da sequência $i, a_i, a_{a_i}, \ldots, 0$ deve ser não superior a H. Estas restrições são complexas de serem consideradas pelos operadores do algoritmo genético. E, se não forem consideradas, um grande número de soluções inviáveis seriam gerados. Assim, em vez de usar essa representação e usar operadores de reparo, é proposta uma representação mais simples, que define apenas o conjunto de *cluster heads*. A floresta de comunicação e a trajetória do agente móvel não são codificados no cromossomo, mas determinadas pelas heurísticas construtivas (apresentadas nas seções anteriores) utilizadas para avaliar a aptidão de uma solução.

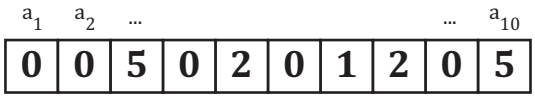


Figura 4.6: Uma possível representação para o cromossomo do Algorítmo Genético.

Assim, o principal objetivo do AG proposto é determinar quais nós sensores devem ser *cluster heads*. Um cromossomo é representado por um *string* de bits, cada gene repre-

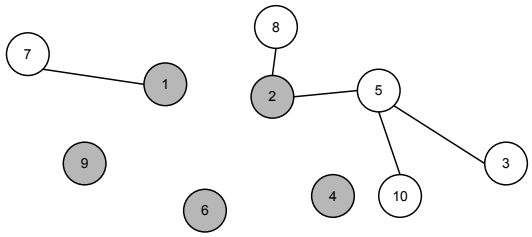


Figura 4.7: Árvore de comunicação codificada no cromossomo da Figura 4.6.

sentando um nó sensor, como na Figura 4.8. Se o gene relacionado a um determinado nó sensor assumir valor 1, este sensor deve ser um *cluster head* nesta solução e portanto deve ser visitado pelo agente móvel. Caso contrário (gene é 0), o sensor deve ser um membro de alguma árvore de comunicação e precisa se conectar a algum *cluster head* usando até H saltos. O primeiro bit corresponde à estação base e sempre recebe valor 1 (definido como *cluster head*). O *fitness* para cada cromossomo é dado pela energia total consumida pela rede, dada pela função (3.6) e penalizada quando inviável, seja pela restrição de saltos ou pela restrição de tempo de percurso da rota do AM. Este valor é calculado pela heurística construtiva descrita na Seção 4.1.

1	0	1	0	0	0	1	1	0	1

Figura 4.8: Representação de um cromossomo.

A população inicial do AG contém P_{size} indivíduos aleatoriamente gerados, os quais são modificados pelos operadores *seleção*, *crossover* e *mutação* gerando sucessivas populações. No fim, o melhor cromossomo (o que contém a melhor avaliação) é retornado, como ilustrado pelo Algoritmo 1.

Algoritmo 1 Pseudo-codigo básico do Algoritmo Genético

```
1: procedure AlgoritmoGenetico(P_size, p_m)
2:     P ← GeraPopulacaoInicial(P_size)
3:     AvaliaPopulacao(P, P_size)
4:     s* ← MelhorIndividuo(P)
5:     while not critério de parada do
6:         P^1 ← Selecao(P)
7:         P^2 ← Crossover(P^1)
8:         P^3 ← Mutacao(P^2, p_m)
9:         P ← P^3
10:        AvaliaPopulacao(P, P_size)
11:        s ← MelhorIndividuo(P)
12:        if f(s) < f(s*) then
13:            s* ← s
14:        end if
15:    end while
16:    return s*
17: end procedure
```

Seleção

A fase de *seleção* é responsável por selecionar os indivíduos que participarão do processo de geração de novos indivíduos. Um dos tipos de seleção é o torneio. Neste processo N indivíduos são selecionados da população atual produzindo um subconjunto de cromossomos. O melhor indivíduo deste subconjunto é então escolhido e adicionado à nova população, a qual será recombinada via crossover e mutação. Neste trabalho foi usado $N = 3$ (torneio ternário).

Crossover

O operador de cruzamento (crossover) é o componente chave no AG. Ele imita a forma de evolução natural. Algumas técnicas de crossover têm sido propostas, tais como *one-point* (Poli & Langdon, 1998) e *multi-point* (Jong & Spears, 1992) crossover.

Neste trabalho foi usada a abordagem *two-point*. Primeiramente, dois cromossomos, da nova população gerada pela fase de seleção, são aleatoriamente selecionados. Depois dois pontos de cruzamento (*two-point*), dentro do cromossomo, são escolhidos aleatoriamente. Em seguida, todos os genes entre os dois pontos são trocados entres os cromossomos pais, produzindo dois novos filhos com características dos pais, como ilustrado pela Figura 4.9.

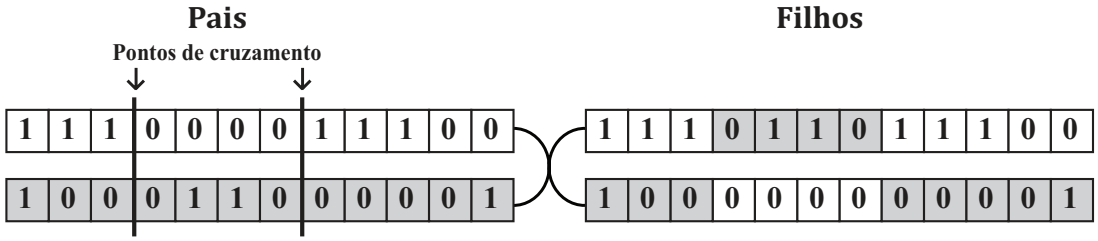


Figura 4.9: Exemplo do operador de cruzamento dois pontos.

Mutação

A mutação é um operador genético que altera um ou mais valores do gene no cromossomo a partir do seu estado inicial. Isto pode resultar em cromossomos com novas características ainda não presentes na população. Com estes novos indivíduos, o algoritmo genético pode ser capaz de chegar a uma solução melhor do que anteriormente era possível. A mutação é uma parte importante da pesquisa genética que pode ajudar a evitar que a população fique presa a um mínimo local. Este operador é aplicado em cada indivíduo com uma certa probabilidade (p_m). Esta probabilidade geralmente deve ser relativamente baixa.

Neste trabalho, o operador de mutação simplesmente inverte um bit aleatoriamente escolhido, ou seja, se o gene do bit é "1", ele é trocado para "0" e vice versa. A Figura 4.10 ilustra esta fase.

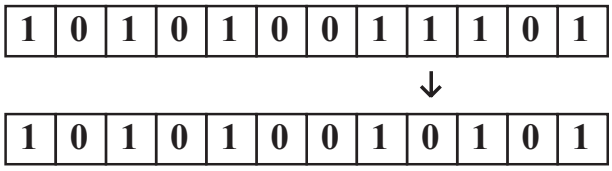


Figura 4.10: Operador de Mutação.

4.3.2 GRASP

A metaheurística *GRASP (Greedy Randomized Adaptive Search Procedure)*, proposta por Feo & Resende (1989), é um processo iterativo *multi-start*, no qual cada iteração consiste em duas fases: a construção de uma solução semi-gulosa (gulosa aleatorizada) e um processo de busca local que visa melhorar a solução gerada na fase anterior. Cada iteração do GRASP trabalha independentemente e o resultado final é a melhor solução encontrada entre todas as iterações. O pseudo-código no Algoritmo 2 ilustra os principais componentes de um GRASP básico, no qual $maxIteration$ interações são executadas. A fase construtiva, incluindo o parâmetro α, é detalhada a seguir. Já o processo de busca local é descrita na Seção 4.4.1.

Algoritmo 2 Pseudo-code básico do GRASP

```
1: procedure GRASP(α,maxIteration)
2:     for i ← 1, maxIteration do
3:         Solucao ← GREEDYRANDOMIZEDCONSTRUCTION(α);
4:         Solucao ← BUSCALOCAL(Solucao);
5:         ATUALIZASOLUCAO(Solucao, MelhorSolucao);
6:     end for
7:     return MelhorSolucao;
8: end procedure
```

Fase Construtiva

A fase de construção do GRASP proposto é um processo iterativo que estima o ganho ao tornar o sensor comum, que não é *cluster head*, em um *cluster head*. O ganho é calculado pela heurística construtiva para a floresta de comunicação (detalhada na Seção 4.1). Se algum nó sensor ficar desconectado na árvore de comunicação, seja devido ao raio de comunicação ou à restrição de saltos, a solução recebe uma penalidade, que é o número de sensores desconectados multiplicado por uma constante grande. Então, os sensores são ordenados por ordem decrescente, de acordo com o valor obtido na etapa anterior e colocados em uma *lista de candidatos (LC)*. Através de um fator α ($0 < \alpha \leq 1$) uma *lista restrita de candidatos (LRC)* é feita, que possui os $max(\alpha \times |LC|, 1)$ melhores elementos de LC. Note que se α for perto de 1, a solução será totalmente aleatória. Por outro lado, se α for próximo de 0 a solução será totalmente gulosa. O próximo nó sensor que se tornará *cluster head* na solução parcial é selecionado aleatoriamente entre os sensores de LRC.

Esta fase é realizada enquanto o tempo gasto na rota entre os *cluster heads* selecionados for menor que T_{max}.

Geralmente, a solução encontrada na fase construtiva não é um mínimo local, então uma fase de busca local tenta melhorar a solução construída. Esta fase é descrita na Seção 4.4.1.

4.4 Busca Local

O objetivo da *Busca Local (BL)* é melhorar a solução previamente encontrada. Os algoritmos de busca local iniciam a partir de uma determinada solução e, em seguida, move-se de forma iterativa a uma solução vizinha. Neste trabalho, são apresentadas três buscas locais, uma para os *cluster heads* selecionados, uma para a árvore de comunicação e outra para a rota do agente móvel, sendo as duas primeiras propostas especificamente para o problema tratado neste trabalho. Todas três usam a abordagem proposta por Araujo *et al.* (2009), onde existe uma lista ordenada que mantém *listLength* soluções potenciais. Uma solução é adicionada e marcada, então enumera-se toda sua vizinhança, mantendo as *listLength* melhores soluções entre as soluções que já estão na lista e os vizinhos da solução que está sendo analisada. Para cada solução não marcada na lista, esta é marcada e são enumerados seus vizinhos. Este processo é repetido enquanto houver solução não marcada na lista. Assim, não apenas o melhor vizinho é considerado, mas também qualquer outro que ainda esteja presente na lista, que pode ser expandida em futuras iterações, evitando dessa forma mínimos locais (Araujo *et al.*, 2009).

As Figuras 4.11 e 4.12 ilustram o processo de busca local com lista de tamanho 3 ($listLength = 3$) aplicado para melhorar a solução da rota do agente móvel. A coluna *Solução* indica a ordem de visita do agente móvel, juntamente com a distância ou tempo da rota (coluna *Custo*) e se a solução já foi expandida ou não (coluna M). Na Figura 4.11, a solução inicial é marcada e seus melhores vizinhos são mantidos na lista, ordenados pelo custo.

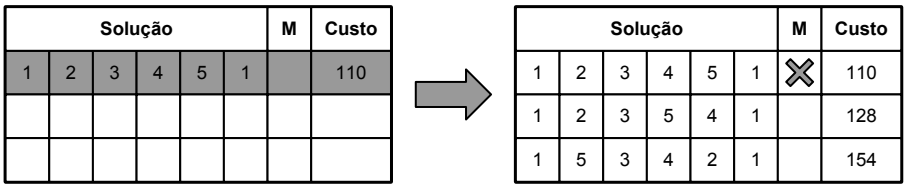


Figura 4.11: Início da Busca Local com lista proposto por Araujo *et al.* (2009).

A Figura 4.12 mostra a continuação do processo de busca. A melhor solução ainda não marcada é marcada e seus vizinhos são enumerados, e os melhores entre esses e os já presentes na lista são mantidos. O processo continua enquanto houver solução cuja vizinhança não tenha sido expandida. Observe que a solução que está sendo expandida

pode ou não continuar na lista, dependendo do custo das soluções geradas. Por exemplo, as soluções de custo 105 e 110 continuaram na lista após serem expandidas. Já a de custo 143 foi substituída por vizinhos com custo menor. Ao fim, a melhor solução é retornada.

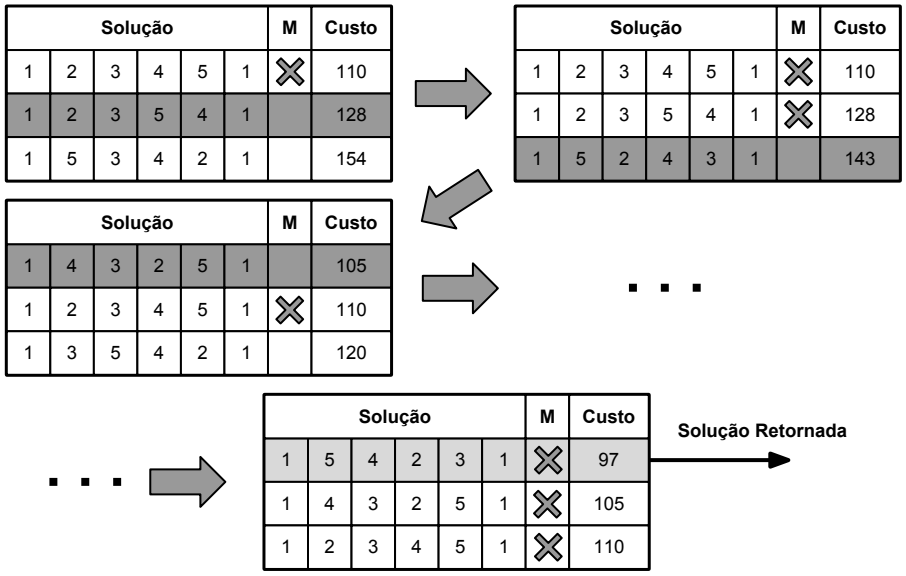


Figura 4.12: Continuação da Busca Local com lista proposto por Araujo *et al.* (2009).

4.4.1 Busca Local para os *Cluster Heads* (BL_{CH})

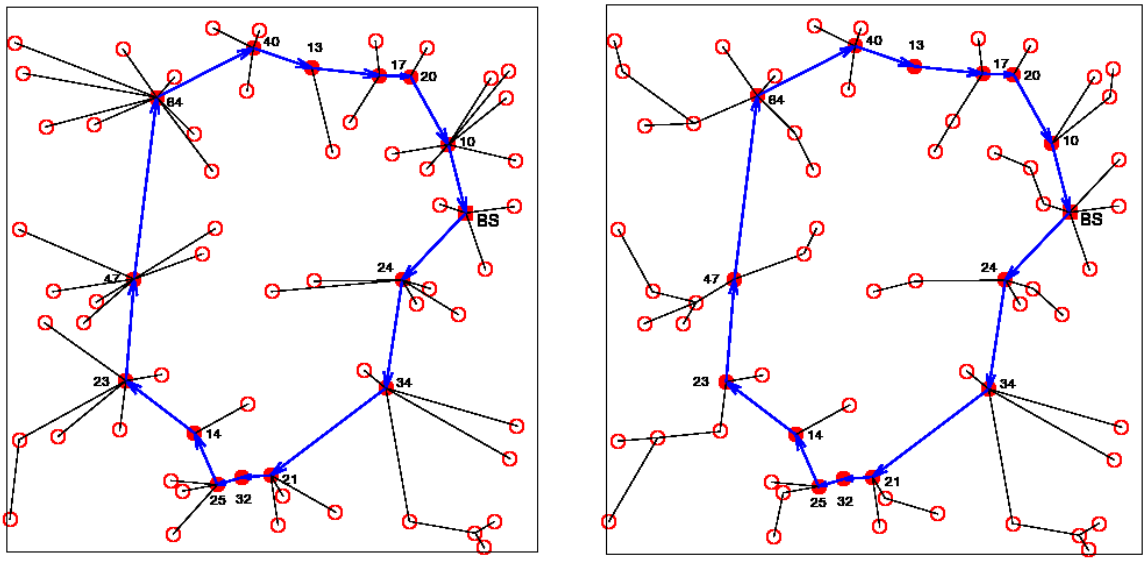
Esta busca local utiliza duas vizinhanças: (*i*) *swap* e (*ii*) *change*. A primeira verifica se a troca do status entre um *cluster head* e um nó sensor comum (o primeiro se torna um sensor comum e o outro um *cluster head*) melhora a qualidade da solução. A fim de reduzir o tamanho da vizinhança, um *cluster head* $c \in \mathcal{S}$ é trocado com um sensor comum $s \in \mathcal{S}$ somente se $d_{cs} \leq R^c$. Se nenhuma melhoria ocorrer com a *swap*, a *change* tenta mudar o status de um único sensor, isto é, se um sensor é *cluster head* ele se torna um sensor comum (não *cluster head*) e vice-versa.

4.4.2 Busca Local para a rede de comunicação (BL_{RC})

Com o objetivo de reduzir o consumo de energia da floresta de comunicação, esta busca local troca as conexões na floresta de comunicação. Para cada sensor i que não é um *cluster head* (isto é, que se comunica com um sensor j), e para cada sensor $k \neq j$ no raio de transmissão de sensor i, um vizinho potencial é aquele que troca a conexão (i,j) para (i,k). Um vizinho é considerado somente se não violar o número máximo de saltos e não deixar um ou mais sensores desconectados de um *cluster head*.

BL_{RC} é importante neste contexto pois a energia consumida é proporcional à quantidade de dados transferidos e à distância que estes dados são enviados. Se um certo sensor i é o mais próximo de vários outros sensores, embora se gaste menos energia para transferir

a i ao invés de transferir para outro sensor, i gastará muita energia para transferir todos os dados recebidos, e a RSSF pode morrer mais rápido. Além disso, como a energia é proporcional a distância ao quadrado (quando a distância entre os sensores é menor que $d_{crossover}$), em alguns casos é melhor usar saltos do que enviar diretamente. A Figura 4.13 mostra o resultado da aplicação da busca local BL_{RC} aplicada a uma floresta de comunicação que foi construída usando o método *H-passos*. Como pode ser percebido, vários sensores passaram a enviar seus dados a outros sensores comuns ao invés de enviar diretamente para o *cluster head*; e outros passaram a enviar diretamente para outro *cluster head*.

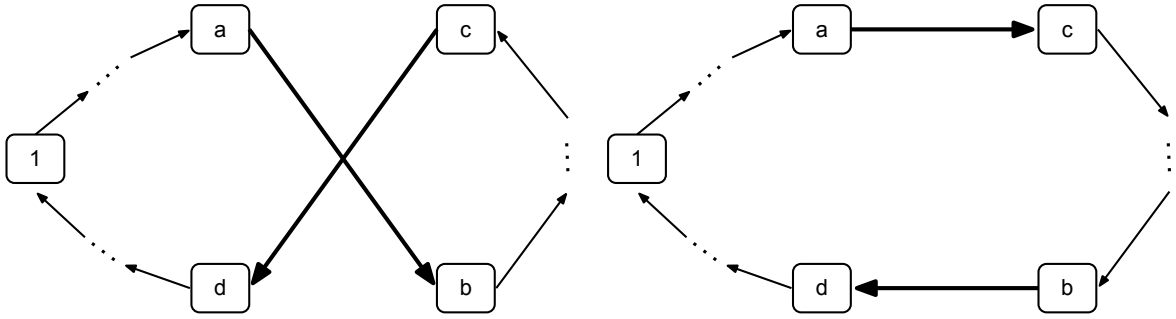


(a) Solução antes da BL_{RC}. (b) Solução depois da BL_{RC}.

Figura 4.13: Busca local BL_{RC} aplicada a uma rede de comunicação.

4.4.3 Busca Local para a rota do agente móvel (BL_{AM})

Esta busca local tem o objetivo de reduzir o tamanho e consequentemente o tempo da rota para um determinado conjunto de *cluster heads*. Para isso, a vizinhança *2-Opt*, proposta por Croes (1958), é usada, trocando dois dos arcos na rota por dois outros arcos. Mais precisamente, em cada passo o algoritmo 2-Opt seleciona duas arestas (a, b) e (c, d) da rota, como $\cdots \rightarrow a \rightarrow b \rightarrow \cdots \rightarrow c \rightarrow d \rightarrow \cdots$, tais que a, b, c, d aparecem nesta ordem na rota e são diferentes, e substitui estes arcos pelos arcos (a, c) e (b, d) originando a rota $\cdots \rightarrow a \rightarrow c \rightarrow \cdots \rightarrow b \rightarrow d \rightarrow \cdots$, desde que essa alteração diminua o comprimento da trajetória. Este movimento é chamado de *troca de 2 elementos*, o qual é ilustrado pela Figura 4.14. Note que os nós entre b e c passam a ser visitados em direção oposta.



(a) Rota antes do 2-Opt (b) Rota depois do 2-Opt

Figura 4.14: Exemplo de uma *troca de 2 elementos* na busca 2-Opt.

4.5 Pseudo-código final das Metaheurísticas

Depois de apresentar os métodos construtivos e as buscas locais que fazem parte da construção e avaliação de cada solução encontrada pelo Algoritmo Genético e GRASP, pode-se exibir os pseudo-códigos das metaheurísticas propostas.

4.5.1 Algoritmo Genético

O procedimento híbrido do Algoritmo Genético está resumido como pseudo-código no Algoritmo 3. O critério de parada é quando o AG tem GWI gerações sem melhora, isto é, quando a melhor solução não muda depois de GWI gerações. Se o AG tem $\frac{GWI}{2}$ gerações sem mudança, a busca local BL_{CH} é aplicada na população, com o objetivo de forçar a população a sair de um possível mínimo local. Esta busca local não é aplicada em cada geração devido ao alto custo computacional. Por este mesmo motivo, a busca local BL_{RC} só é aplicada na melhor solução encontrada no fim do AG, reavaliando sua função de avaliação. Além disso, em nossos experimentos, aplicar a BL_{RC} em cada indivíduo a cada geração não resultou em melhoras significativas, quando comparados com o acréscimo de tempo ao adicionar a busca. Como a busca local BL_{AM} é muito rápida, ela é aplicada na avaliação de cada indivíduo, como mostra o Algoritmo 4. Vale ressaltar que, com exceção da BL_{CH}, as buscas locais são aplicadas somente para melhorar a avaliação da solução, melhorando a floresta de comunicação e o tamanho da rota do agente móvel, não para melhorar a seleção dos *cluster heads*, isto é feito exclusivamente pelo AG e pela BL_{CH}.

4.5.2 GRASP

O pseudo-código do Algoritmo 5 resume o procedimento híbrido do GRASP. O critério de parada é o número máximo de interações ($maxIteration$). O Algoritmo 6 ilustra a heurística construtiva e as buscas locais propostas para avaliar a solução. Observe que no GRASP, em cada solução gerada, as três buscas locais propostas são aplicadas, diferentemente do Algoritmo Genético.

Algoritmo 3 Pseudo-código final do Algoritmo Genético

```
1: procedure AlgoritmoGenetico(n, d_{n×n}, P_size, H, p_m, GWI)
2:     P ← GeraPopulacaoInicial(n, P_size)
3:     AvaliaPopulacao(P, n, d_{n×n}, P_size, H, D_max)
4:     s* ←MelhorIndividuo(P)
5:     gwi ← 0
6:     while gwi ≤ GWI do
7:         P^1 ← Selecao(P)
8:         P^2 ← Crossover(P^1)
9:         P^3 ← Mutacao(P^2, p_m)
10:        P ← P^3
11:        AvaliaPopulacao(P, n, d_{n×n}, P_size, H, D_max)
12:        s ← MelhorIndividuo(P)
13:        if f(s) < f(s*) then
14:            s* ← s
15:            gwi ← 0
16:        else
17:            gwi ← gwi + 1
18:        end if
19:        if gwi = GWI/2 then
20:            BL_CH(P)
21:        end if
22:    end while
23:    BL_RC(s*)
24:    return s*
25: end procedure
```

Algoritmo 4 Pseudo-código da função de avaliação do AG

```
1: procedure AvaliaPopulacao(P, n, d_{n×n}, P_size, H, D_max)
2:     for i ← 0 to P_size do
3:         // ** Constrói a rede de comunicação e a rota do AM **
4:         P[i].Solution ← HeuristicaConstrutiva(P[i].CHSelected, n, d_{n×n}, H, D_max);
5:         P[i].Solution ← BL_AM(P[i].Solution);
6:         P[i].E_c ← EnergiaConsumida(P[i].Solution, n, d_{n×n})
7:     end for
8: end procedure
```

Algoritmo 5 Pseudo-código final do GRASP

```
1: procedure Grasp(n, d_{n×n}, H, D_max, maxIteration, α)
2:     for i ← 0 to maxIteration do
3:         CHSelected ← GreedyRandomizedConstruction(α);
4:         CHSelected ← BL_CH(CHSelected);
5:         s ← AvaliaSolução(CHSelected, n, d_{n×n}, H, D_max)
6:         if i = 0 then
7:             s* ← s
8:         else if f(s) < f(s*) then
9:             s* ← s
10:        end if
11:    end for
12:    return s*
13: end procedure
```

Algoritmo 6 Pseudo-código da função de avaliação do GRASP

```
1: procedure AvaliaSolução(CHSelected, n, d_{n×n}, H, D_max)
2:     // ** Constrói a rede de comunicação e a rota do AM **
3:     Solucao ← HeuristicaConstrutiva(CHSelected, n, d_{n×n}, H, D_max);
4:     Solucao ← BL_RC(Solucao);
5:     Solucao ← BL_AM(Solucao);
6:     return Solucao
7: end procedure
```

4.6 Comentários finais

Neste capítulo foram apresentadas as heurísticas construtivas e metaheurísticas auxiliadas por busca locais para construir e avaliar as solução construídas por cada método. Um Algoritmo Genético e um GRASP foram propostos para tentar encontrar o melhor conjunto de *cluster heads*.

No próximo capítulo são apresentados os resultados dos experimentos computacionais encontrados pela formulação matemática do problema e pelas heurísticas.

Capítulo 5

Resultados Computacionais

Neste capítulo, são apresentados os experimentos computacionais conduzidos para avaliar e comparar os resultados obtidos pelos métodos híbridos com um método exato, que resolve a formulação PLIM do PARST. O propósito é validar os algoritmos híbridos para serem utilizados em um contexto de RSSF.

Visando avaliar os algoritmos propostos, são realizados testes envolvendo um conjunto de instâncias nas quais as posições dos nós sensores, exceto a estação base, são aleatoriamente geradas sobre uma área quadrada plana de lado $L = 500m$. A estação base foi posicionada no centro da rede, ou seja, na posição (250, 250). Para balancear o consumo de energia com o atraso na entrega dos dados, o tempo máximo gasto pelo agente móvel na rota foi limitado em $T_{max} = \{0, 120, 300, 600, \infty\}$ segundos, de forma que $T_{max} = 0$ significa que o agente móvel não será utilizado e apenas a estação base atuará como *cluster head*. Em outras palavras, o que importa, nesse caso, não é o consumo de energia e sim o atraso na entrega dos dados. Quando $T_{max} = \infty$ o agente móvel pode gastar o tempo que for necessário na rota, ou seja, o atraso não importa e sim o consumo de energia.

A Tabela 5.1 lista os valores dos parâmetros relacionados com os dados de entrada. Os cinco últimos estão relacionados com o consumo de energia e são os mesmo utilizados por Heinzelman *et al.* (2002).

A Tabela 5.2 lista os parâmetros do Algoritmo Genético. Um teste de análise de variância (ANOVA) foi realizado com um conjunto de valores para cada parâmetro e um subconjunto de instâncias. A diferença não foi estatisticamente significativa para qualquer um deles, então foi decidido usar a combinação que gerou a melhor média e o menor desvio padrão. Estes são os valores indicados na tabela.

A fim de definir os parâmetros do GRASP (α e $maxIteration$) também foram realizados alguns testes estatísticos, usando a análise de variância (ANOVA), com algumas instâncias. Para isso, o GRASP foi executado 10 vezes para cada combinação dos parâmetros. Os resultados são resumidos no gráfico da Figura 5.1, que mostra a média da energia consumida para o conjunto de instâncias para algumas combinações dos parâmetros, onde, `Ixxx_Ayy` significa $maxIteration =$ `xxx` e $\alpha =$ `yy` (em porcentagem). Como

Tabela 5.1: Dados dos parâmetros relacionados e seus valores

Descrição	**Valor**
Área da rede	$500 \times 500m^2$
Número de nós sensores	{20, 30, 40, 50, 60, 70, 80, 100}
Tempo máximo da rota do agente móvel (T_{max})	{0, 120, 300, 600, ∞} seg
Raio de comunicação dos sensores (R^c)	250m
Número máximo de saltos (H)	3
Altura do AM sobre a área de sensoriamento (H_{AM})	50m
Tamanho dos dados sensoriados (k)	80bits
Velocidade do agente móvel (v_{am})	5m/s
Tempo de parada do AM em cada *cluster head*	5seg
Constante da eletrônica para transmitir e receber dados (E_{elec})	50nJ/bit
Atenuação do sinal espaço livre (ϵ_{fs})	10pJ/bit/m^2
Atenuação do sinal multicaminho (ϵ_{mp})	0,0013pJ/bit/m^4
Energia inicial dos nós sensores (E_{ini})	0,5J
Distância para a troca entre os modelos de atenuação ($d_{crossover}$)	87m

Tabela 5.2: Parâmetros do algoritmo genético e seus valores

Descrição	**Valor**
Gerações sem melhora (GWI)	20
Tamanho da população inicial (P_{size})	150
Taxa de mutação (P_m)	7%
Tamanho da lista da Busca Local ($listLength$)	3

pode ser observado, embora não exista diferença estatística, independentemente do valor de $maxIteration$, o melhor resultado foi obtido com α (em porcentagem) igual a 20%. A diferença entre `I150_A20` e `I200_A20` é bastante sutil, mas o tempo para executar 150 iterações é inferior a 200 iterações. Por estas razões foi escolhido $\alpha = 20\%$ e $maxIteration = 150$ para serem usados no algoritmo GRASP. Como no Algoritmo Genético, o tamanho da lista da Busca Local ($listLength$) foi definido como 3, como mostra a Tabala 5.3.

Tabela 5.3: Parâmetros do GRASP e seus valores

Descrição	**Valor**
Número de iterações ($maxIteration$)	150
Alfa (α)	20%
Tamanho da lista da Busca Local ($listLength$)	3

Como foram definidos dois métodos construtivos para construir a floresta de comunicação (*H-passos* e *Prim-H*), é preciso definir qual se comporta melhor, gerando soluções de melhor qualidade, em conjunto com a busca local BL_{RC}. Para isso, alguns testes foram realizados considerando $H = 3$ e $T_{max} = 0$. A Tabela 5.4 mostra os resultados obtidos considerando todas as instâncias testadas nos experimentos. Como pode ser percebido, o método *H-passos* obteve melhores resultados em quatro instâncias e para as outras duas o resultados foi o mesmo alcançado pelo método *Prim-H*. Por esse motivo, mesmo o tempo sendo um pouco maior, o método *H-passos* será usado como heurística construtiva para a floresta de comunicação, uma vez que ele se comportou melhor em conjunto com a BL_{RC}.

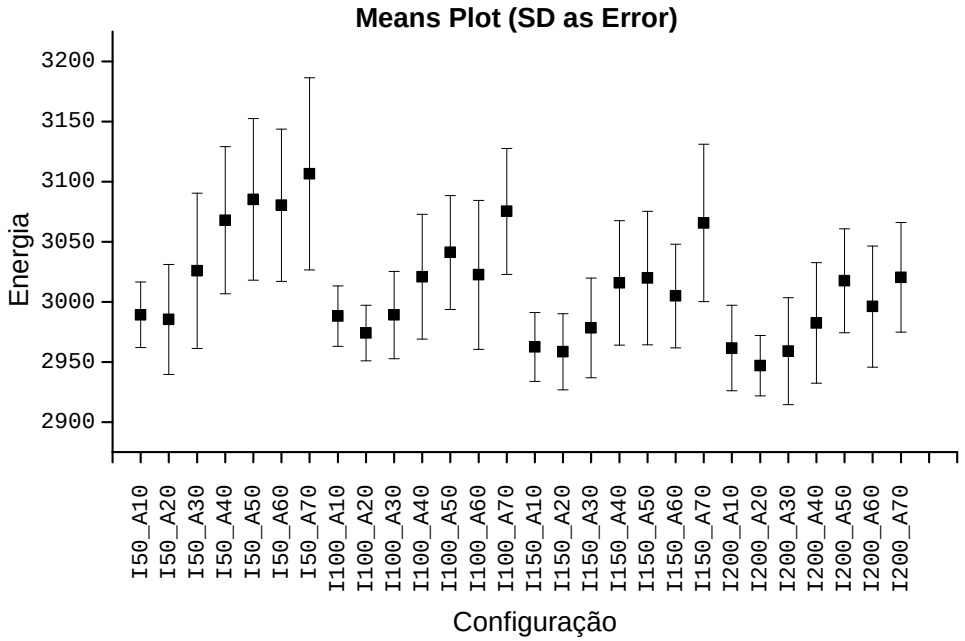


Figura 5.1: Resultado da análise de variância para diferentes configurações do GRASP.

Tabela 5.4: Comparação entre os métodos heurísticos *H-passos* e *Prim-H*

Sensores	***H-passos***		***Prim-H***	
	Energia(μJ)	**Tempo(s)**	**Energia(μJ)**	**Tempo(s)**
20	1220,19	0,01	1220,19	0,00
30	2108,87	0,02	2108,87	0,01
40	**2215,55**	0,04	2422,17	0,03
60	**1930,17**	0,23	2079,02	0,13
80	**2929,52**	0,54	3101,96	0,39
100	**3381,54**	1,42	3407,65	0,91

O PARST foi resolvido de duas formas, pelo método exato e pelos métodos híbridos. Todos os algoritmos foram implementados em C++ e compilados com GNU g++ versão 4.3.4. Como método exato para encontrar a solução ótima, a formulação matemática do PARST foi resolvida pelo ILOG CPLEX versão 12.5, sendo implementada via ILOG Concert Technology. O tempo de relógio foi limitado em 12 horas para o algoritmo exato. Todos os experimentos foram realizados em um máquina Intel Xeon 2.67GHz, com 24 GB de RAM, com Sistema Operacional Linux.

Para estimar o quão longe a solução encontrada pelo método exato após 12 horas está da solução ótima, recupera-se o *gap* de dualidade para cada instância. Esse *gap* corresponde à diferença percentual entre um limite inferior da solução do problema e a melhor solução inteira viável obtida pelo método exato.

5.1 Método exato

A Tabela 5.5 apresenta os resultados computacionais associados ao método exato. As três primeiras colunas são relacionadas à instância: número de sensores, tempo limite da rota, e nome da instância, a qual é nomeada usando os dados das colunas anteriores. As próximas três colunas estão relacionadas à solução do método exato: energia consumida pela melhor solução encontrada, tempo de execução, e o *gap* de dualidade obtido pelo método quando finalizado. Nesta tabela, as entradas "-" significam que o algoritmo exato não foi capaz de encontrar uma solução viável para o PARST, dentro do limite de 12 horas, para a instância associada àquela linha na tabela.

Os resultados obtidos indicam que é difícil resolver o PARST através de um algoritmo exato baseado na formulação matemática proposta, principalmente para instâncias maiores. Para instâncias pequenas, com 20 ou 30 sensores, o algoritmo foi capaz de encontrar 9 soluções ótimas em menos de 12 horas. Na instância `N40-T0` a memória disponível não foi suficiente para o método exato, por isso o algoritmo foi finalizado antes do tempo limite de execução. Para instâncias com mais de 60 nós sensores o método foi capaz de encontrar uma solução viável apenas quando $T_{max} = 0$, ou seja, quando não há agente móvel.

Tabela 5.5: Resultado do Algoritmo Exato

n	T_{max}	**ID**	**Método Exato**		
			$E_c(\mu\mathbf{J})$	**Tempo(s)**	**Gap(%)**
20	0	N20-T0	1220,19	58	0,00
	120	N20-T120	852,27	1393	0,00
	300	N20-T300	327,94	327	0,00
	600	N20-T600	114,00	20	0,00
	∞	N20-T∞	114,00	24	0,00
30	0	N30-T0	2108,87	7001	0,00
	120	N30-T120	1247,88	†	36,42
	300	N30-T300	418,08	7141	0,00
	600	N30-T600	181,42	1622	0,00
	∞	N30-T∞	172,43	1337	0,00
40	0	N40-T0	1939,79$^{\triangleleft}$	18217	34,32
	120	N40-T120	1060,37	†	28,43
	300	N40-T300	562,48	†	16,15
	600	N40-T600	266,83	27237	0,00
	∞	N40-T∞	234,00	24603	0,00
60	0	N60-T0	1889,60	†	29,95
	120	N60-T120	-	-	-
	300	N60-T300	-	-	-
	600	N60-T600	-	-	-
	∞	N60-T∞	-	-	-
80	0	N80-T0	3023,83	†	46,55
	120	N80-T120	-	-	-
	300	N80-T300	-	-	-
	600	N80-T600	-	-	-
	∞	N80-T∞	-	-	-
100	0	N100-T0	4265,03	†	55,49
	120	N100-T120	-	-	-
	300	N100-T300	-	-	-
	600	N100-T600	-	-	-
	∞	N100-T∞	-	-	-

† *Tempo limite de 12 horas excedido*
◁ *Memória insuficiente (Out of memory)*

Com o objetivo de melhorar o resultado do método exato, a solução viável obtida pelo

método híbrido será utilizada como uma solução inicial para o algoritmo exato. Dessa forma, o método exato inicia com uma solução viável e tenta melhorá-la, diminuir o *gap*, ou provar a otimalidade dentro do limite de tempo.

5.2 Algoritmo Genético

Na Tabela 5.6 são apresentados os resultados computacionais associados ao algoritmo genético e ao método exato, que agora usa a melhor solução viável obtida pelo algoritmo genético como um solução inicial. As três primeiras colunas são relacionadas à instância: número de sensores, tempo limite da rota e nome da instância. As próximas três estão relacionadas ao algoritmo genético: o tempo gasto para 10 execuções do algoritmo genético, a energia total gasta pela melhor solução encontrada, e o desvio padrão em relação a energia consumida para as 10 execuções. Soluções marcados com '*' são soluções ótimas. As colunas restantes estão relacionadas com a solução exata do PARST: tempo de execução, energia consumida pela melhor solução encontrada, a melhora em relação à solução inicial dada pelo AG, e o *gap* de dualidade reportado pelo método exato quando finalizado. As entradas marcadas com "-" indicam que a medida vale 0.

Em primeiro lugar, perceba que a melhoria alcançada pela solução exata sobre as soluções dadas pelo AG, em sua maioria, são notavelmente baixas. Para as instâncias pequenas isto acontece porque o AG é capaz de encontrar a solução ótima ou uma próxima da ótima. Para instâncias maiores, isso acontece porque o algoritmo exato não conseguiu melhorar a solução inicial (encontrada pelo AG) após 12h. Em ambos os casos mostram a eficiência do AG proposto. Cada caso é discutido a seguir.

Para instâncias pequenas, com 20 ou 30 nós sensores, o método exato provou a otimalidade em 9 casos. Em 7 deles, ambos os métodos encontraram a solução ótima: `N20-T0`, `N20-T300`, `N20-T600`, `N20-T∞`, `N30-T0`, `N30-T600` e `N30-T∞`. Em outras duas (`N20-T120` e `N30-T300`) o algoritmo exato conseguiu encontrar a solução ótima depois de melhorar 0,79% e 0,16% respectivamente a solução obtida pelo AG. Particularmente, para a instância `N30-T300`, o tempo total de execução do AG foi de aproximadamente 74 segundos, enquanto o método exato gastou mais de 5 horas para melhorar menos de 0.2% e provar a otimalidade da solução. A instância `N30-T120` foi a única, no grupo com 20 e 30 sensores, em que o algoritmo exato não conseguiu encontrar a solução ótima em 12 horas, tendo um *gap* de 31,11%, e melhorando apenas 1,75% a solução inicial dada pelo AG. Observe que o tempo gasto pelo AG para encontrar uma solução dessa instância foi de apenas 13 segundos. Isto sugere que o AG proposto juntamente com as heurísticas construtivas são aptos para encontrar boas soluções (e possivelmente ótimas), e as buscas locais utilizadas são capazes de encontrar um mínimo local (ou possivelmente global) para instâncias em que o espaço de busca é relativamente pequeno.

Para instâncias de tamanho médio (com 40 ou 60 nós sensores) o algoritmo exato foi

executado pelas 12 horas permitidas e apenas uma solução ótima foi encontrada (N40-T∞). Nas outras, se a solução ótima foi encontrada, a otimalidade não foi provada. O tempo de execução do AG para a instância N40-T∞ foi de aproximadamente 73 segundos, enquanto o método exato demorou mais de 10 horas para provar a otimalidade da solução obtida pelo AG. Para as instâncias N40-T300 e N40-T600 o *gap* foi de menos de 15%. Entretanto, o tempo gasto nesta instâncias pelo algoritmo exato foi de 12 horas, enquanto o AG gastou menos de 2 minutos e a solução encontrada pelo AG não foi melhorada pelo algoritmo exato. O maior ganho pelo método exato, em relação a solução do AG, foi alcançado na instância N40-T120, mas ele precisou do tempo limite de 12 horas e ainda teve um *gap* maior que 30%. Já para a instância N40-T0 o método exato conseguiu um ganho de 10% sobre a solução inicial, mas foi interrompido antes das 12 horas por falta de memória. Todas os casos de teste com 60 nós sensores tiveram um gap maior que 30% e apenas em N60-T0 o algoritmo exato conseguiu melhorar a solução do AG, mas gastando um tempo extremamente superior.

Finalmente, para as instâncias de tamanho grande (com 80 ou 100 nós sensores) a formulação exata não foi útil em praticamente nenhum caso, conseguindo melhorar a solução encontrada pelo genético apenas para a instância N80-T0. Para as outras instâncias, depois de 12 horas de execução, o método exato não encontrou nenhuma nova solução e o *gap* ainda permaneceu alto.

O gráfico da Figura 5.2 mostra o consumo de energia total encontrado pelo algoritmo genético para cada instância, deixando claro que, independentemente do número de sensores, quanto maior for o valor de T_{max}, menor o consumo de energia, e vice versa.

5.3 GRASP

Na Tabela 5.7 são apresentados os resultados obtidos pelo algoritmo GRASP proposto e a solução encontrada pelo método exato baseada na solução encontrada pelo GRASP. As três primeiras colunas listam o número de sensores, tempo limite da rota e nome da instância. As próximas duas colunas mostram os resultados do algoritmo GRASP: tempo de execução em segundos e o consumo de energia da melhor solução encontrada. Soluções marcados com '*' são soluções ótimas. As colunas restantes estão relacionadas com a formulação exata: tempo de execução, melhor solução encontrada (usando a solução obtida pelo GRASP como ponto inicial), a melhora percentual sobre a solução inicial, e o *gap* de dualidade reportado pelo CPLEX. As entradas marcadas com "-" indicam que a medida vale 0.

Observe que, para o conjunto de instâncias pequenas (20 e 30 nó sensores), o algoritmo GRASP encontrou 8 das 9 soluções ótimas. A otimalidade destas instâncias foram provadas pelo método exato, atingindo um gap de 0% sem nenhuma melhora na solução. Isto sugere que, como no AG, o GRASP proposto e as heurísticas construtivas são capazes de identificar

Tabela 5.6: Desempenho do Algoritmo Genético proposto para diferentes cenários

n	T_{max}	ID	Algoritmo Genético			Método Exato			
			Tempo(s)	$E_c(\mu J)$	DP(%)	Tempo(s)	$E_c(\mu J)$	Ganho(%)	Gap(%)
20	0	N20-T0	0	1220,19*	0,0	29	1220,19	-	-
	120	N20-T120	3	858,99	0,0	560	852,27	0,79	-
	300	N20-T300	4	327,94*	4,9	207	327,94	-	-
	600	N20-T600	6	114,00*	14,0	13	114,00	-	-
	∞	N20-T∞	9	114,00*	14,0	11	114,00	-	-
30	0	N30-T0	0	2108,87*	0,8	14282	2108,87	-	-
	120	N30-T120	13	1269,74	1,3	†	1247,88	1,75	31,11
	300	N30-T300	74	418,77	3,8	19352	418,08	0,16	-
	600	N30-T600	32	181,42*	8,8	735	181,42	-	-
	∞	N30-T∞	22	172,43*	9,2	132	172,43	-	-
40	0	N40-T0	0	2147,44	0,7	21688◁	1939,79	10,70	32,53
	120	N40-T120	38	1259,96	1,3	†	1052,31	19,73	30,89
	300	N40-T300	35	560,53	2,8	†	560,53	-	14,14
	600	N40-T600	116	273,22	5,8	†	273,22	-	5,09
	∞	N40-T∞	73	234,00*	6,8	37068	234,00	-	-
60	0	N60-T0	2	1930,17	0,8	†	1888,60	2,20	30,28
	120	N60-T120	1016	1490,52	1,1	†	1490,52	-	58,90
	300	N60-T300	139	881,28	1,8	†	881,28	-	34,40
	600	N60-T600	373	523,88	3,0	†	523,88	-	54,67
	∞	N60-T∞	435	350,20	4,6	†	350,20	-	32,19
80	0	N80-T0	6	2988,38	0,5	†	2890,03	3,40	42,92
	120	N80-T120	588	2018,74	0,8	†	2018,74	-	79,63
	300	N80-T300	249	1271,16	1,3	†	1271,16	-	38,97
	600	N80-T600	1280	844,19	1,9	†	844,19	-	62,15
	∞	N80-T∞	1212	472,13	3,4	†	472,13	-	32,73
100	0	N100-T0	16	3290,45	0,5	†	3290,45	-	41,56
	120	N100-T120	1443	2599,32	0,6	†	2599,32	-	79,76
	300	N100-T300	781	1642,95	1,0	†	1642,95	-	71,60
	600	N100-T600	1558	1144,94	1,4	†	1144,94	-	65,10
	∞	N100-T∞	1692	592,12	2,7	†	592,12	-	32,84

† *Tempo limite de 12 horas excedido*
◁ *Memória insuficiente (Out of memory)*

boas soluções (e ótimas). Note que para as instâncias `N30-T0` e `N30-T300` a solução ótima foi encontrada em poucos segundos pelo GRASP, mas o algoritmo exato usou quase 2 horas para provar a otimalidade de cada uma. E para a instância `N30-T120` uma solução foi encontrada em menos de 1 segundo pelo GRASP e o exato conseguiu melhorá-la em aproximadamente 8%, mas gastou as 12 horas limites.

Para as instâncias de tamanho médio (40 e 60 nós sensores) nenhuma solução ótima foi provada. No entanto, o ganho obtido pelo método exato na solução encontrada pelo GRASP foi muito pequeno, exceto para as instâncias `N40-T0` e `N40-T120`, onde o ganho foi de aproximadamente 10% e 19%, respectivamente. Observe que, novamente, para a instância `N40-T0`, o algoritmo exato foi finalizado pela falta de memória e não pelo tempo limite do algoritmo. Para as instâncias grandes (80 e 100 sensores), não se pode confirmar a qualidade das soluções obtidas pelo GRASP, uma vez que o *gap* de dualidade foi, em sua maioria, acima de 40%, mas o método ainda é muito útil. O tempo de execução foi muito baixo, oscilando de 5 segundos a 14 minutos, enquanto método exato não conseguiu fazer qualquer melhora em 12 horas.

A Figura 5.3 apresenta a solução encontrada pelo GRASP para a instância com 100 sensores para os diferentes valores de T_{max}. Quando o valor de T_{max} é zero (Figura 5.3a) todos os nós sensores devem enviar seus dados para a estação base, e então muitos sensores

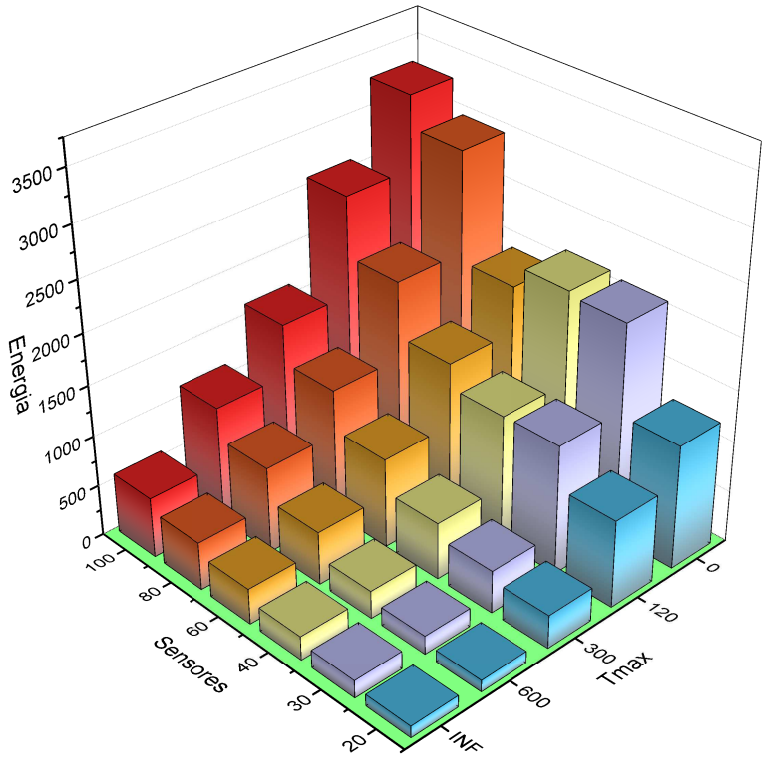


Figura 5.2: Gráfico da energia total gasta em cada instância pela solução encontrada pelo Algoritmo Genético.

usam os 3 saltos permitidos para transmitir seus dados, aumentando, assim, o consumo de energia, mas o atraso na entrega dos dados é mínimo. Novamente, muitos sensores usam os 3 saltos quando $T_{max} = 120$ (Figura 5.3b), mas agora outros sensores se tornaram *cluster head*. Aumentado o valor de T_{max} (Figura 5.3c e Figura 5.3d) o número de *cluster heads* também aumenta e a maioria dos nós sensores comuns conseguem transmitir diretamente para algum *cluster head*, diminuindo, assim, a energia gasta. Quando T_{max} é ilimitado (Figura 5.3e) praticamente todos os nós sensores se tornam *cluster head* (apenas um permaneceu como sensor comum, já que a distância entre ele o *cluster head* mais próximo é menor que H_{AM}). Neste caso, o consumo de energia é mínimo, mas o atraso na entrega dos dados é máximo. Pode-se observar na Tabela 5.7, que quanto maior o tempo de percurso do AM maior será o tempo de execução do GRASP, o que sugere que as heurísticas têm mais dificuldade na resolução de casos com maior número de *cluster heads*. Neste caso, acredita-se que a busca local BL_{CH} (e provavelmente BL_{AM}) é a mais demorada. A busca local BL_{RC}, ao contrário, trabalha mais quando T_{max} é pequeno, mas ela não parece afetar significativamente o tempo computacional.

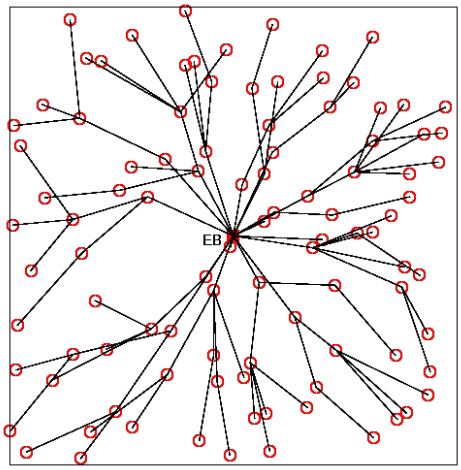
Tabela 5.7: Desempenho do GRASP proposto para diferentes cenários

n	T_{max}	ID	GRASP		Solução Exata			
			Tempo(s)	$E_c(\mu J)$	Tempo(s)	$E_c(\mu J)$	Ganho(%)	Gap(%)
20	0	N20-T0	0	1220,19*	32	1220,19	-	-
	120	N20-T120	0	900,67	838	852,27	5,68	-
	300	N20-T300	1	327,94*	163	327,94	-	-
	600	N20-T600	1	114,00*	11	114,00	-	-
	∞	N20-T∞	1	114,00*	11	114,00	-	-
30	0	N30-T0	0	2108,87*	6611	2108,87	-	-
	120	N30-T120	0	1344,65	†	1247,88	7,75	28,29
	300	N30-T300	2	418,08*	6388	418,08	-	-
	600	N30-T600	9	181,42*	488	181,42	-	-
	∞	N30-T∞	7	172,43*	127	172,43	-	-
40	0	N40-T0	0	2147,44	22341$^{\triangleleft}$	1939,79	10,70	32,34
	120	N40-T120	4	1259,96	†	1052,31	19,73	30,40
	300	N40-T300	5	569,04	†	561,42	1,36	12,07
	600	N40-T600	28	281,07	†	274,90	2,24	6,19
	∞	N40-T∞	21	234,00	†	234,00	-	32,48
60	0	N60-T0	1	1930,17	†	1888,60	2,20	31,55
	120	N60-T120	18	1540,92	†	1540,92	-	60,20
	300	N60-T300	21	889,45	†	889,45	-	34,46
	600	N60-T600	640	542,65	†	542,65	-	56,24
	∞	N60-T∞	106	350,20	†	350,20	-	32,19
80	0	N80-T0	6	2988,38	†	2890,03	3,40	42,92
	120	N80-T120	36	2018,74	†	2018,74	-	57,63
	300	N80-T300	46	1272,89	†	1272,89	-	39,05
	600	N80-T600	157	915,23	†	915,23	-	65,09
	∞	N80-T∞	325	472,13	†	472,13	-	32,73
100	0	N100-T0	15	3290,45	†	3290,45	-	42,29
	120	N100-T120	207	2599,32	†	2599,32	-	79,76
	300	N100-T300	107	1648,79	†	1648,79	-	71,70
	600	N100-T600	341	1159,53	†	1159,53	-	65,70
	∞	N100-T∞	812	592,12	†	592,12	-	32,84

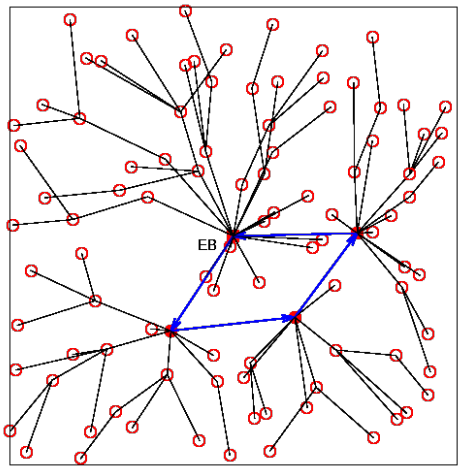
† *Tempo limite de 12 horas excedido*
◁ *Memória insuficiente (Out of memory)*

5.4 Algoritmo Genético vs. GRASP

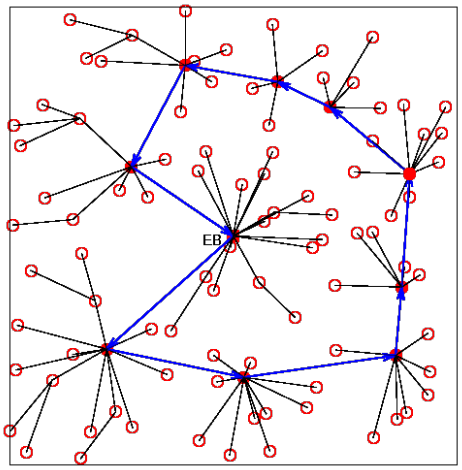
O gráfico da Figura 5.4 apresenta uma comparação entre as soluções encontrada pelo algoritmo genético e pelo GRASP. As barras da esquerda representam, em porcentagem, o quão melhor é a solução obtida pelo AG em comparação com a o GRASP. Já as barras da direita mostram o quão mais rápido é o GRASP se comparado com o AG. Analisando as barras da esquerda, em 18 instâncias os dois algoritmos encontraram as mesmas soluções. Em 11 casos o AG conseguiu obter uma solução melhor do que o GRASP, que conseguiu apenas uma solução melhor que o AG (instância `N30-T300`). Entretanto, ao se comparar o AG com o GRASP em relação ao tempo, o GRASP é muito mais rápido, chegando a ser mais de 55 vezes mais rápido que o AG para a instância `N60-T120`. Apenas na instância `N60-T600` o AG foi um pouco mais eficiente. É importante lembrar que o algoritmo genético é executado 10 vezes e o tempo exibido é a soma das 10 execuções, ou seja, o tempo de cada execução é na verdade quase 10 vezes menor.



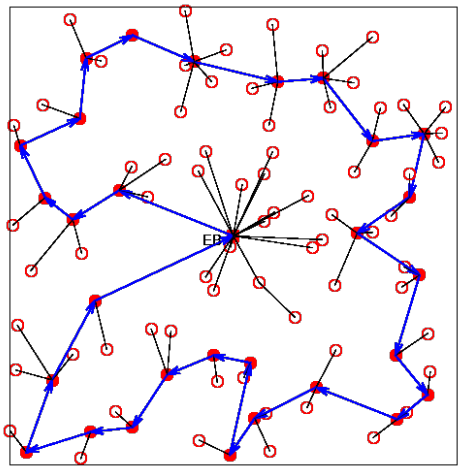
(a) $T_{max} = 0$ $E_c = 3290{,}5\mu$J



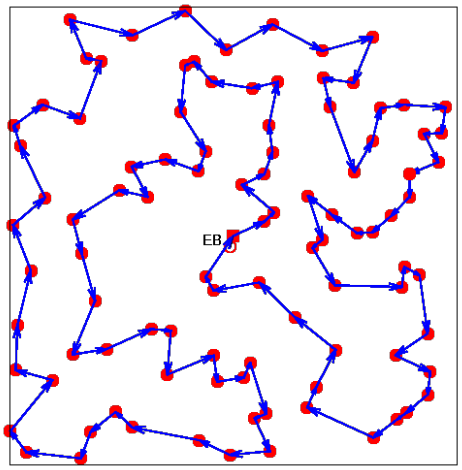
(b) $T_{max} = 120$ $E_c = 2599{,}3\mu$J



(c) $T_{max} = 300$ $E_c = 1648{,}8\mu$J



(d) $T_{max} = 600$ $E_c = 1159{,}5\mu$J



(e) $T_{max} = \infty$ $E_c = 592{,}1\mu$J

Figura 5.3: Soluções encontrada pelo GRASP para a instância com 100 sensores para os diferentes valores de T_{max}.

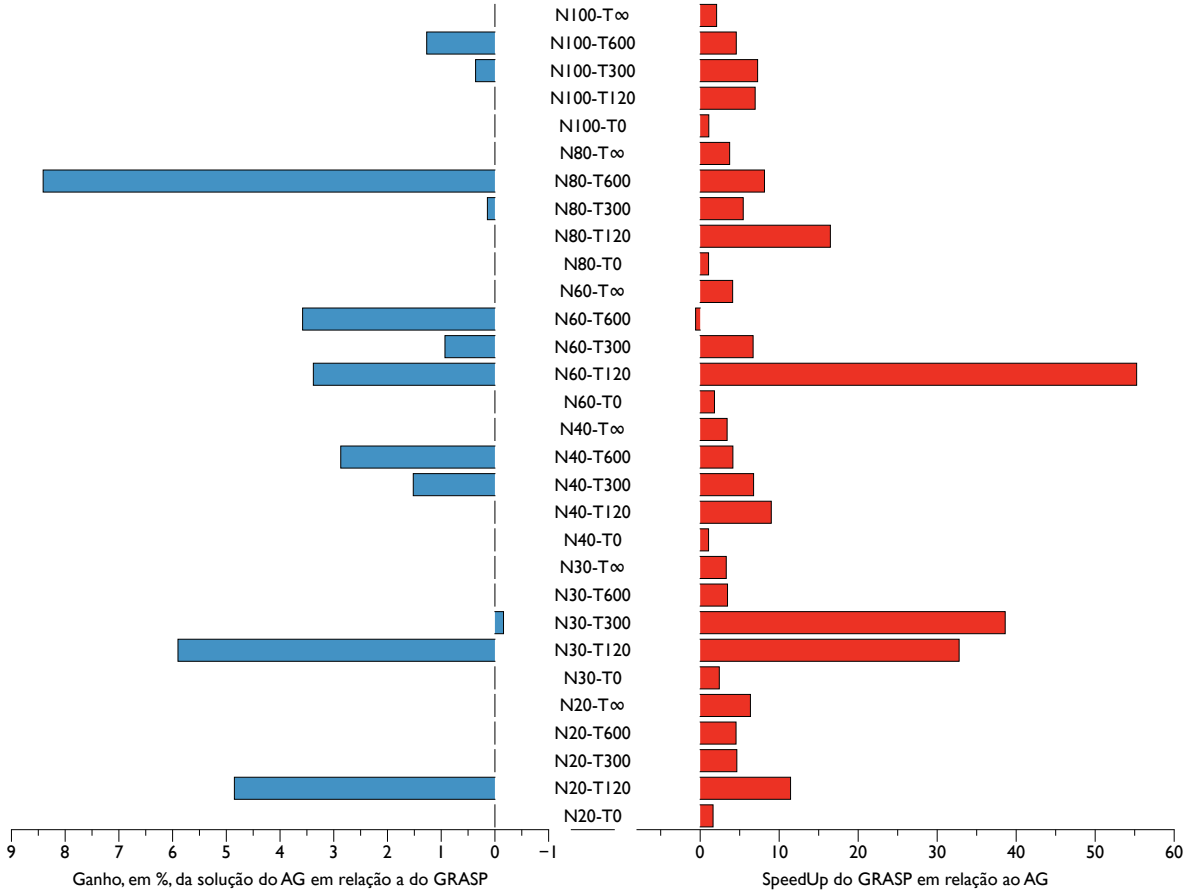


Figura 5.4: Comparação entre as soluções obtidas pelo Algoritmo Genético e pelo GRASP.

5.5 Maximizando o tempo de vida da RSSF

Como visto no Seção 3.4, uma abordagem para tentar aumentar o tempo de vida da RSSF é utilizar não apenas uma rede configuração (de energia e translação), mas várias. Isso pode ser feito por geração de colunas, resolvendo-se o PARST várias vezes, utilizando os preços duais. Entretanto, como foi mostrado nas tabelas das seções anteriores, usar o modelo exato para o PARST demanda muito tempo. Dessa forma, usar na geração de colunas o modelo exato para o subproblema demoraria muito tempo (semanas) para se chegar à solução final. Assim, é apropriado resolver o subproblema pelo algoritmo genético ou pelo GRASP, e quando estes não conseguirem gerar uma coluna (configuração) de custo reduzido positivo, o subproblema é revolvido de forma exata (modelo SUB_{RSSF}).

Assim, como no subproblema SUB_{RSSF}, onde a solução é guiada pelos preços duais fornecidos pelo problema mestre, no algoritmo genético e no GRASP também deve-se utilizá-los com o objetivo de guiar a construção da solução. Para isso, é preciso mudar a função objetivo incorporando os preços duais: a energia gasta por cada nó sensor é multiplicada pelo seu preço dual. Este processo também é aplicado na busca local BL_{RC}. Como o AG e o GRASP geram várias soluções, ao invés de se retornar a melhor solução é

retornado um conjunto com até dez soluções viáveis com custo reduzido positivo. Dessa forma, o número de vezes que o problema mestre é resolvido pode ser reduzido, já que várias colunas são adicionadas de uma vez. Apesar de aumentar o tamanho do problema mestre mais rapidamente, muitas vezes com configurações que não farão parte da solução final, antecipa-se a inclusão de algumas colunas que poderiam ser necessárias futuramente no problema mestre (Santos, 2008).

Mesmo o subproblema exato sendo chamado poucas vezes, o tempo gasto por ele, em cada chamada, é muito alto para que ele encontre a solução ótima para o problema. Por isso, o tempo de execução do SUB_{RSSF} foi limitado: se ele conseguir uma solução de custo reduzido positivo dentro do tempo limite a geração de colunas continua, caso contrário ela é encerrada e a solução inteira do PM_{RSSF} corrente é retornada.

É fácil perceber que, quanto maior o tempo limite, melhor será a solução encontrada pelo SUB_{RSSF} e melhor será a solução encontrada pela geração de colunas. Entretanto, o tempo total da geração de colunas também será grande. Por outro lado, se o tempo limite for pequeno, o problema SUB_{RSSF} será interrompido com uma solução não muito boa, e assim tende a ser chamado mais vezes. Além disso, a solução final da geração de colunas não será a melhor possível, ganhando apenas em tempo de execução.

Como dito na Seção 3.4, neste trabalho é adotado como tempo de vida da rede o número de *rounds* (número de vezes) que o agente móvel pode percorrer sua trajetória antes que o primeiro sensor esgote sua energia por completo. Assim, maximizar o tempo de vida da rede é o mesmo que maximizar o número de rounds. Se for utilizada apenas uma configuração de rede, o número de *rounds* é dado pela divisão da energia inicial do nó sensor E_{ini} pela energia gasta do nó sensor com o maior consumo de energia na solução e_m ($e_m \geq Et_i + Er_i$, $\forall i \in \mathcal{S} \backslash \{s_1\}$). Usando várias configurações, é a soma do número de *rounds* que cada uma é utilizada. No problema PM_{RSSF} isso é dado por $\sum_{i \in \mathcal{C}} \lambda_i$. Assim, a função objetivo passa a ser maximizar o número de *rounds*.

Para definir o melhor tempo limite do SUB_{RSSF} foi feita uma análise de variância (ANOVA) para alguns valores, com algumas instâncias. Para tentar melhorar o tempo total e a solução da geração de colunas, alguns dos testes incluem mais de um tempo limite. Assim, se o SUB_{RSSF} encontrar uma solução de custo reduzido positivo dentro do tempo limite inicial, esta solução é retornada, caso contrário, o tempo limite é aumentado e o SUB_{RSSF} continua de onde parou, e assim por diante, até que todos os tempos limites sejam testados. Quando o SUB_{RSSF} não encontrar uma solução de custo reduzido positivo mesmo sem atingir o tempo limite, a geração de colunas termina e a solução inteira do PM_{RSSF} é retornada. A Figura 5.5 ilustra este processo, juntamente com a interação entre o método híbrido e o subproblema.

Os gráficos da Figura 5.6 mostram o resultado da análise de variância usando o Algoritmo Genético, que foi executado 10 vezes para cada combinação, como heurística para o subproblema. Cada configuração possui um ou mais números que representam o

tempo limite, em segundos, para o SUB_{RSSF}. Configurações com mais de um tempo limite significam que o SUB_{RSSF} poderá ser executado mais de uma vez, considerando cada um dos tempos limites, conforme descrito acima. O gráfico da Figura 5.6a mostra que quanto maior o tempo limite maior será o número de *rounds* da solução encontrada pela geração de colunas, mas é maior também o tempo de execução. (Figura 5.6b). Entretanto, se o tempo limite do subproblema for pequeno a geração de colunas não conseguirá obter valores altos para o número de *rounds*, mas o tempo total de execução é pequeno. Como nota-se pela Figura 5.6a, a configuração `300` é estatisticamente melhor, mas é a que gasta o maior tempo. Por outro lado, a configuração `60` é finalizada com o menor tempo, mas é estatisticamente pior que outras configurações em relação ao número de *rounds*. Já as configurações `60_60`, `60_60_180`, `120` e `120_180` são estatisticamente iguais, mas `60_60` gasta, em média, o menor tempo e gerou o menor desvio padrão entre o tempo de execução. Por isso, é usada como tempos limites a configuração `60_60` para o SUB_{RSSF} em conjunto com o AG.

O mesmo teste estatístico é executado considerando o GRASP, que também foi executado 10 vezes para cada combinação, como heurística para o subproblema, como se pode observar nos gráficos da Figura 5.7. As configurações `60` e `60_60` são estatisticamente piores que as outras em relação ao número de *rounds*. Já as configurações `60_60_180`, `120`, `120_180` e `120` são estatisticamente iguais para o número de *rounds*, como a configuração `120` possui, em média, o menor tempo de execução, será usado como tempo limite `120` segundos para o SUB_{RSSF} quando o GRASP for a heurística usada.

A Tabela 5.8 apresenta os resultados obtidos considerando o algoritmo genético como heurística para o subproblema. As três primeiras colunas são relacionadas à instância: número de sensores, tempo limite da rota e nome da instância. As próximas duas colunas mostram o número de *rounds* e o tempo gasto sem a utilização da geração de colunas, ou seja, o número de *rounds* quando se minimiza a energia total consumida, a melhor solução encontrada pelo AG e o tempo gasto nesta execução. Maximizando o número de *rounds* (usando a abordagem de geração de colunas), as próximas cinco colunas mostram o número de *rounds*, o número de colunas adicionadas, o número de iteração entre o problema mestre e o subproblema, o tempo gasto e o *gap* de dualidade retornado pelo PM_{RSSF} ao fim de sua execução. Por fim, a última coluna mostra o ganho ao se maximizar o número de *rounds* da RSSF em relação à abordagem que minimiza o consumo total de energia da rede.

Percebe-se que não houve ganho sempre que $T_{max} = \infty$, pois para este caso o consumo de energia é o mínimo possível, independente do número de sensores. Assim, mesmo que existam configurações diferentes, elas combinadas gerem o mesmo número de *rounds*. Para todos os outros valores de T_{max}, exceto em `N20-T600`, houve um aumento considerável no número de *rounds*, chegando a ser mais de cinco vezes maior na instância `N40-T120`. Também pode-se observar que o tempo de execução para maximizar o número de *rounds*

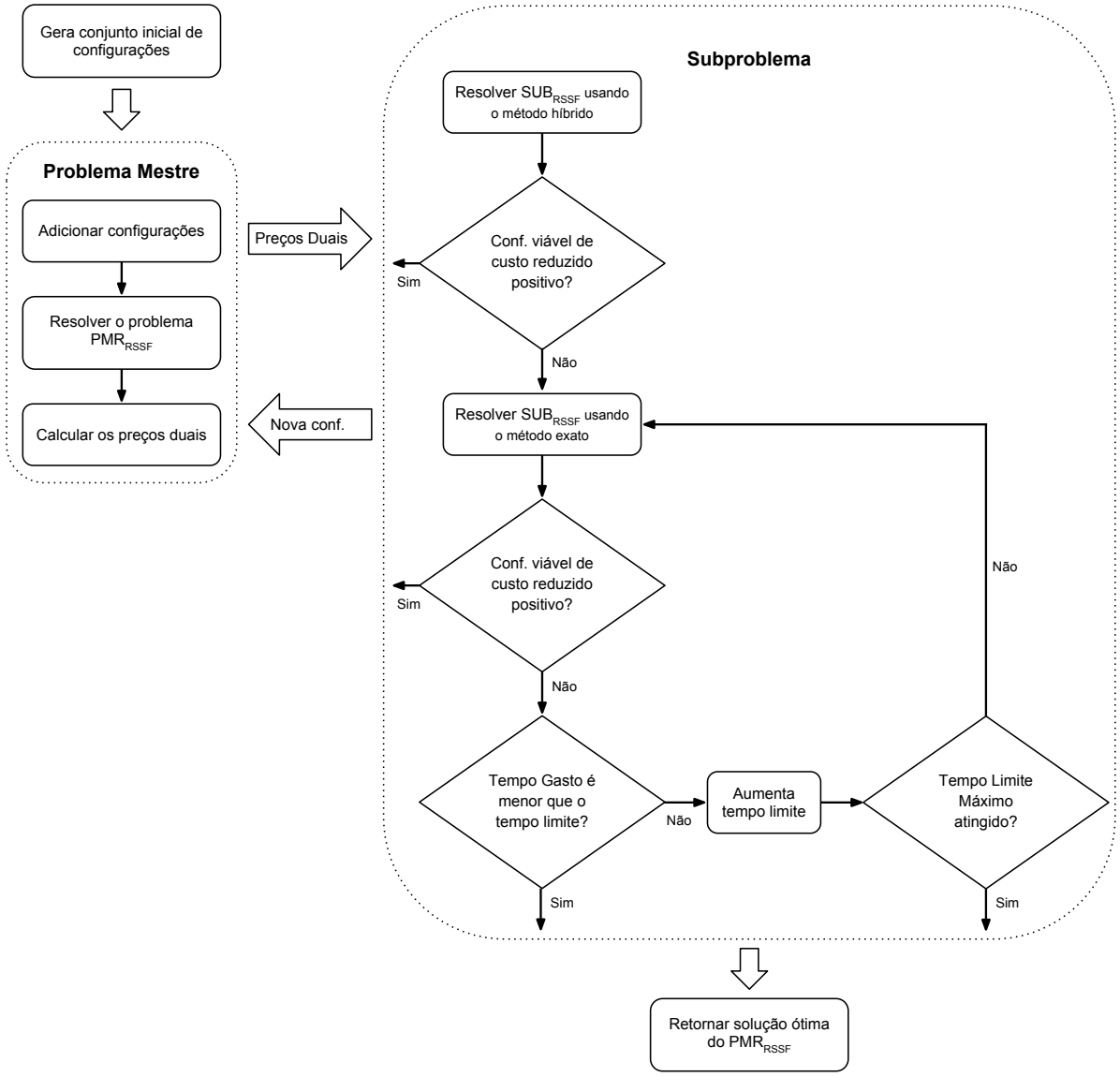
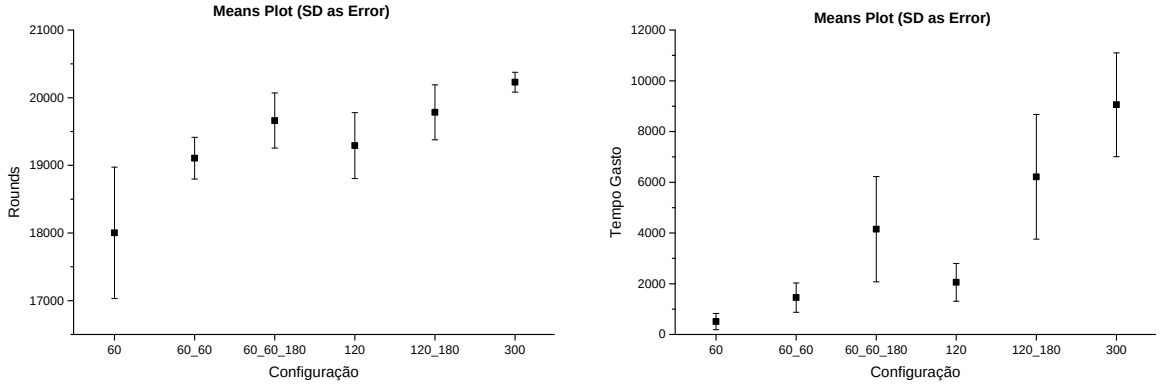


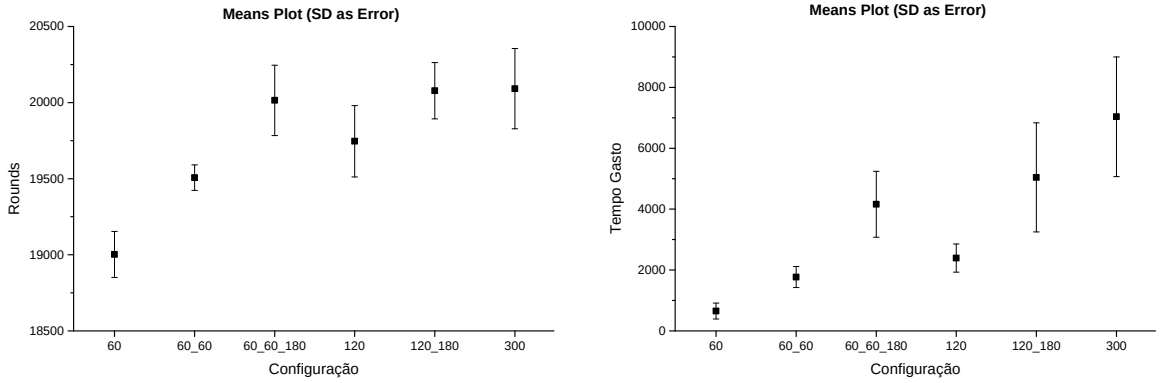
Figura 5.5: Solução heurística do subproblema incorporada no método de geração de colunas, baseado em Santos (2008).

é bem maior, como se esperava. Na maioria das instâncias esse tempo foi bem menor que 1 hora, mas em alguns casos passou de 3 horas de execução (instâncias `N80-T0`, `N80-T120` e `N100-T0`). Para essas instâncias o gargalo estava na resolução da solução inteira do problema PM_{RSSF}, uma vez que o número de iterações, na maioria dos casos, foi pequeno. O problema PM_{RSSF} também foi interrompido antes de encontrar a solução ótima, devido a falta de memória, para as instâncias `N40-T0`, `N40-T120` e `N80-T0`, mostrando o quão complicado é resolver a solução inteira do problema mestre. Para as instâncias `N20-T`∞, `N60-T`∞, `N80-T`∞ e `N100-T`∞ foi adicionada mais de uma coluna e o número de iterações também foi maior que um. Isso ocorre quando existem sensores bastante próximos, onde a distância entre eles é menor que a distância do agente móvel. Assim, sensores muito próximos da estação base podem deixar de ser *cluster heads* e enviarem seus dados diretamente



(a) Número de *rounds* (b) Tempo gasto

Figura 5.6: Análise de variância para diferentes tempos limites para o SUB_{RSSF} em conjunto com o Algoritmo Genético.



(a) Número de *rounds* (b) Tempo gasto

Figura 5.7: Análise de variância para diferentes tempos limites para o SUB_{RSSF} em conjunto com o GRASP.

para a estação base e vice versa, ou sensores bem próximos podem trocar seu status, isto é, em determinada configuração um é *cluster head* e o outro é sensor comum e vice versa.

A Tabela 5.9 apresenta os resultados obtidos considerando o GRASP como heurística para o subproblema. As três primeiras colunas listam o número de sensores, tempo limite da rota, e nome da instância. As próximas duas colunas mostram o número de *rounds* e o tempo gasto sem a utilização da geração de colunas, ou seja, o número de *rounds* considerando a melhor configuração encontrada pelo GRASP em uma execução e o tempo gasto nesta execução. Usando geração de colunas, as próximas cinco colunas mostram o número de *rounds*, o número de colunas adicionadas, o número de iterações entre o problema mestre e o subproblema, o tempo gasto e o *gap* de dualidade retornado pelo PM_{RSSF} ao fim de sua execução. Por fim, a última coluna mostra o ganho ao se maximizar o número de *rounds* da RSSF em relação à abordagem que minimiza o consumo total de energia da rede.

Os resultados obtidos pelo GRASP, no geral, foram muito parecidos com os obtidos

Tabela 5.8: Maximizando o tempo de vida da RSSF com Algoritmo Genético

n	T_{max}	ID	Min. E_c		Max. número de *rounds*					Ganho
			Rounds	Tempo(s)	*Rounds*	#C	#I	Tempo(s)	Gap(%)	
20	0	N20-T0	1747	0,1	3313	10	12	0,5	-	1,90
	120	N20-T120	3208	0,4	3313	3	4	2,4	-	1,03
	300	N20-T300	5678	0,5	19783	204	87	1671,0	-	3,48
	600	N20-T600	83333	0,8	83333	27	13	126,5	-	-
	∞	N20-T∞	83333	0,7	83333	33	25	546,7	-	-
30	0	N30-T0	785	0,2	2812	172	200	2745,8	-	3,58
	120	N30-T120	1907	2,1	5028	80	40	528,8	-	2,64
	300	N30-T300	10869	2,0	26356	344	112	258,0	-	2,42
	600	N30-T600	31250	3,1	55583	173	34	215,1	-	1,78
	∞	N30-T∞	83333	2,6	83333	1	3	11,1	-	-
40	0	N40-T0	1297	0,4	5790$^{\triangleleft}$	311	337	7489,8	0,03	4,46
	120	N40-T120	1852	4,7	9687$^{\triangleleft}$	317	108	3813,4	0,01	5,23
	300	N40-T300	11142	3,4	13521	19	6	131,5	-	1,21
	600	N40-T600	31250	15,0	33302	119	25	298,3	-	1,07
	∞	N40-T∞	83333	7,7	83333	1	3	134,0	-	-
60	0	N60-T0	2115	1,8	3420	34	36	131,3	-	1,62
	120	N60-T120	4310	17,8	7280	210	68	1936,5	-	1,69
	300	N60-T300	5813	11,7	12620	24	6	180,9	-	2,17
	600	N60-T600	13888	35,2	16078	11	4	190,2	-	1,16
	∞	N60-T∞	83333	43,8	83333	2	3	208,2	-	-
80	0	N80-T0	2485	3,5	7155$^{\triangleleft}$	425	427	13530,8	0,08	2,88
	120	N80-T120	2212	41,3	5526	77	23	45128,1	-	2,50
	300	N80-T300	3968	24,5	19742	244	51	2020,4	-	4,98
	600	N80-T600	10869	135,0	40140	667	288	2138,8	-	3,69
	∞	N80-T∞	83333	141,6	83333	59	12	1385,5	-	-
100	0	N100-T0	2156	12,3	4947	603	605	31894,5	-	2,29
	120	N100-T120	2032	133,3	4905	41	12	1935,0	-	2,41
	300	N100-T300	3676	124,7	8388	10	4	373,8	-	2,28
	600	N100-T600	8928	183,9	10618	21	5	437,1	-	1,19
	∞	N100-T∞	83333	327,2	83333	21	5	1216,8	-	-

$\triangleleft$ *Memória insuficiente (Out of memory)*

pelo AG. Pode-se destacar que o maior tempo gasto foi para a instância N40-T0, que gastou aproximadamente 4 horas, enquanto a instância que mais demorou no AG foi a instância N80-T120, que gastou quase 13 horas para ser finalizada. Além disso, com o GRASP, para cinco instâncias (N40-T0, N40-T120, N80-T0, N80-T120 e N80-T300) a memória disponível foi insuficiente para terminar a formulação inteira do problema PM_{RSSF}. Estas foram as instâncias que mais demoraram a ser finalizadas, um indício forte de que o gargalo está na resolução do PM_{RSSF}. Praticamente todas as instâncias tiveram um aumento no número de *rounds*, ao se maximizar o número de *rounds*, chegando a ter um ganho de mais de 4 vezes nas instâncias N40-T0, N40-T120 e N80-T300.

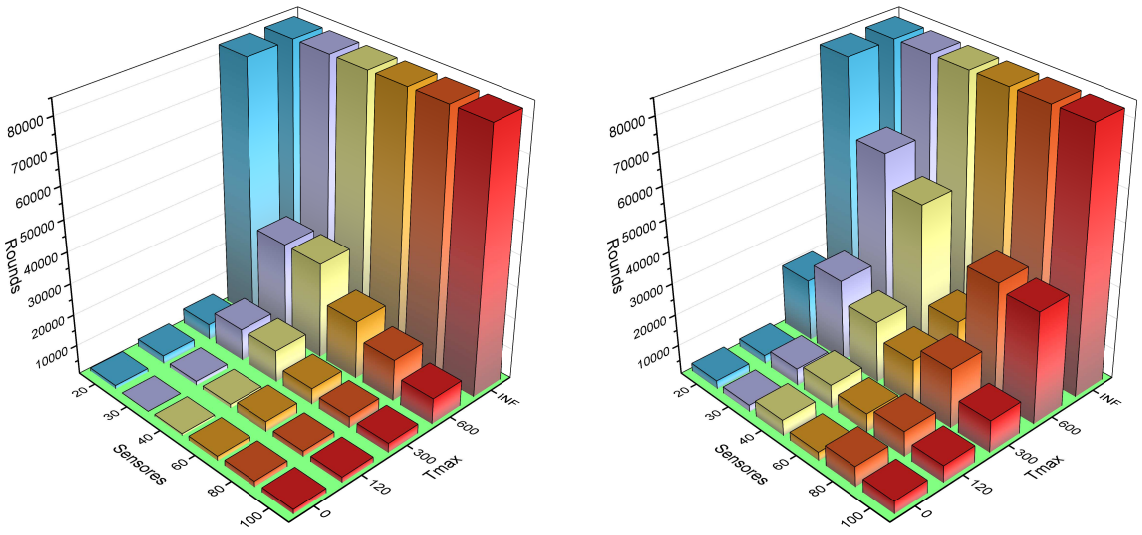
Os gráficos da Figura 5.8 mostram o número de *rounds* sem geração de colunas (Figura 5.8a), minimizando a energia total da rede, e com geração de colunas (Figura 5.8b), maximizando o tempo de vida da rede, quando o GRASP é utilizado como heurística auxiliar para o subproblema. Eles deixam claro o ganho ao se utilizar a abordagem com geração de colunas para aumentar o tempo de vida da rede. Observe que, impendente do número de sensores, o maior aumento está nos valores mais altos de T_{max}.

O gráfico da Figura 5.9 compara o número de *rounds* e o tempo de execução da geração de colunas considerando o AG e o GRASP para cada instância. Ao contrário da Figura 5.4, as colunas do gráfico à esquerda mostram o quão melhor, em porcentagem, é o GRASP em

Tabela 5.9: Maximizando o tempo de vida da RSSF com o GRASP

n	T_{max}	ID	Min. E_c		Max. número de *rounds*					Ganho
			Rounds	Tempo(s)	*Rounds*	#C	#I	Tempo(s)	Gap(%)	
20	0	N20-T0	1747	0,1	3313	10	12	0,8	-	1,90
	120	N20-T120	3208	0,3	3313	2	4	2,8	-	1,03
	300	N20-T300	5678	1,3	20419	427	93	3389,0	-	3,60
	600	N20-T600	83333	1,4	83333	30	32	227,9	-	-
	∞	N20-T∞	83333	1,4	83333	39	41	250,3	-	-
30	0	N30-T0	785	0,3	2800	173	175	2918,7	-	3,57
	120	N30-T120	1945	0,6	6085	95	58	248,4	-	3,13
	300	N30-T300	10869	2,0	26788	591	64	271,5	-	2,46
	600	N30-T600	31250	8,9	59811	23	9	19,9	-	1,91
	∞	N30-T∞	83333	6,7	83333	1	3	16,0	-	-
40	0	N40-T0	1297	0,6	5710$^{\triangleleft}$	234	236	15034,3	0,02	4,40
	120	N40-T120	1852	4,6	8155$^{\triangleleft}$	218	65	8314,0	0,03	4,40
	300	N40-T300	10869	5,6	20158	59	8	164,3	-	1,85
	600	N40-T600	31250	27,6	49814	46	7	280,8	-	1,59
	∞	N40-T∞	83333	19,5	83333	1	3	162,4	-	-
60	0	N60-T0	2115	2,5	3420	34	36	147,9	-	1,62
	120	N60-T120	3968	19,0	6512	15	4	168,7	-	1,64
	300	N60-T300	5813	22,4	14996	40	6	239,5	-	2,58
	600	N60-T600	19230	88,7	19230	10	3	294,0	-	-
	∞	N60-T∞	83333	94,3	83333	2	3	33,9	-	-
80	0	N80-T0	2485	8,3	7155$^{\triangleleft}$	425	427	14779,8	0,08	2,88
	120	N80-T120	2427	31,4	8748$^{\triangleleft}$	262	51	12412,5	0,04	3,60
	300	N80-T300	4310	48,8	19560$^{\triangleleft}$	243	40	11815,0	0,01	4,54
	600	N80-T600	13888	163,8	38200	298	192	2905,5	-	2,75
	∞	N80-T∞	83333	305,9	83333	41	38	2276,3	-	-
100	0	N100-T0	2156	20,8	4801	300	302	1955,6	-	2,23
	120	N100-T120	2427	192,8	6087	110	17	10124,0	-	2,51
	300	N100-T300	3676	113,0	10914	86	11	1143,2	-	2,97
	600	N100-T600	8928	358,4	20468	50	7	1492,8	-	2,29
	∞	N100-T∞	83333	725,4	83333	94	92	4119,1	-	-

$\triangleleft$ *Memória insuficiente (Out of memory)*



(a) Min. energia consumida total (b) Max. número de *rounds*

Figura 5.8: Número de *rounds* ao se minimizar a energia total consumida e ao se maximizar o número de rouds utilizando o GRASP como heurística para o subproblema.

relação ao AG para o número de *rounds* obtido. Já as colunas à direita mostram o quão mais rápido é a execução do AG, em porcentagem, em relação ao GRASP. Primeiramente,

observe que na maioria das instâncias o GRASP consegue obter uma solução com mais *rounds* do que o AG, chegando a ser mais de 90% maior para a instância `N100-T600`, mas para esta instância o GRASP chegou a ser quase 250% mais lento que o AG. Em alguns poucos casos o GRASP foi pior, chegando a obter quase 20% menos *rounds* que o AG, como para a instância `N40-T120`. O AG se mostrou, na maioria dos casos, mais rápido do que o GRASP, em especial, para a instância `N80-T300` foi quase 500% mais rápido e na `N100-T120` o AG foi mais de 400% mais rápido. Para alguns poucos casos, o AG se mostrou mais lento que o GRASP, por exemplo, para a instância `N100-T0` o GRASP gastou aproximadamente 95% a menos de tempo.

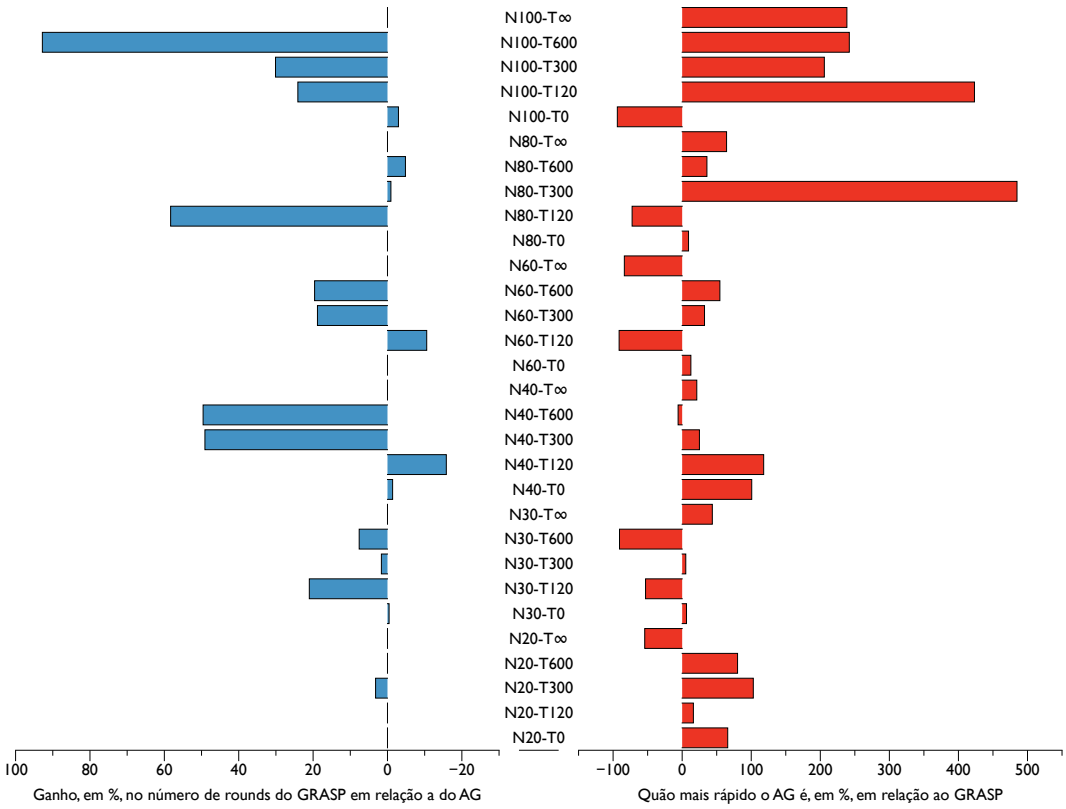


Figura 5.9: Comparação para o número de *rounds* e o tempo da geração de colunas considerando o Algoritmo Genético e o GRASP.

Para mostrar a rápida aproximação da solução ótima pelos métodos híbridos, a geração de colunas foi executada com a formulação exata do subproblema (SUB_{RSSF}) sem limite de tempo e sem solução inicial (gerada pelo AG ou GRASP), para a instância `N20-T300`. Foi escolhida uma instância com 20 sensores pois o SUB_{RSSF} (sem limite de tempo e sem solução inicial) consegue ser finalizado em tempo hábil apenas para essa quantidade. A escolha de $T_{max} = 300$ foi feita porque nesse caso a geração de colunas com solução heurística do SUB_{RSSF} não atinge a mesma solução quando o SUB_{RSSF} é resolvido de forma exata. Limitando-se o tempo total de execução em 1 hora, tem-se o gráfico da Figura 5.10. Note que o método exato demora quase 4 minutos para encontrar sua primeira

solução, com aproximadamente 5000 *rounds*, enquanto as soluções dos métodos híbridos, em 4 minutos, já conta com quase 20000 *rounds*. O AG é o primeiro a terminar, gastando quase 30 minutos. Entretanto, o AG encontra uma solução um pouco pior que o GRASP, que termina em aproximadamente 55 minutos. Observe que em 1 hora de execução o método exato conseguiu uma solução com bem menos de 20000 *rounds*, mostrando o quão demorada é sua execução.

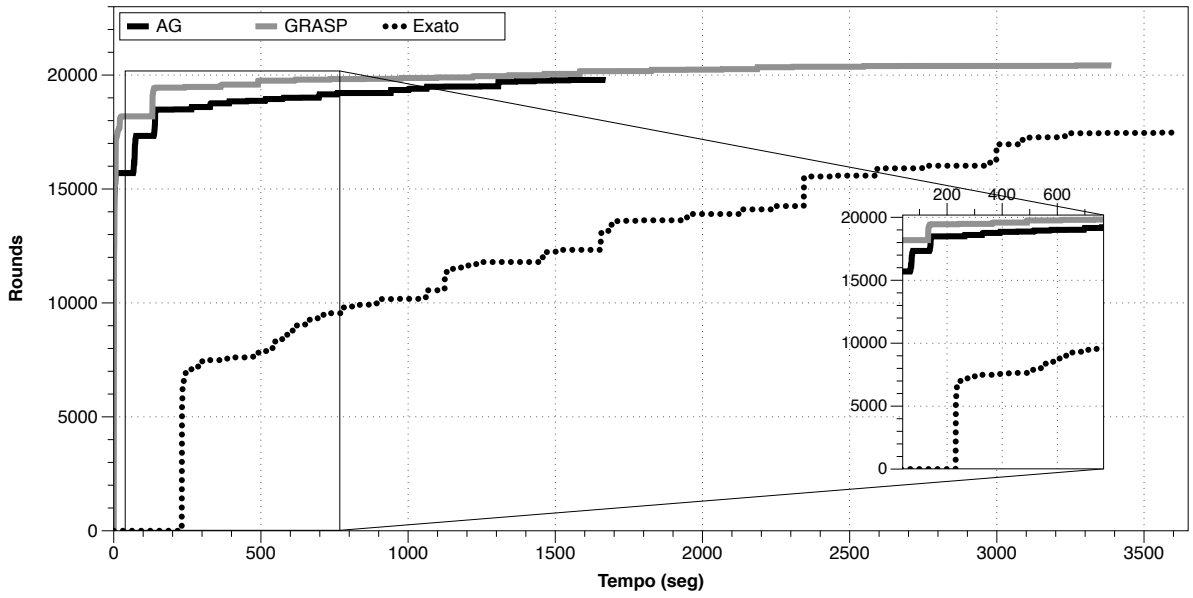


Figura 5.10: Comparação do valor da solução com o tempo de execução para a instância `N20-T300`.

Capítulo 6

Considerações Finais

6.1 Conclusões

Neste trabalho é apresentada uma abordagem para diminuir o consumo de energia em Rede de Sensores Sem Fio, utilizando um agente móvel. Esta abordagem considera que o gasto de energia é proporcional à quantidade de dados (pacotes) recebidos e transmitidos por cada nó sensor nas árvores de comunicação. O problema pode ser dividido em encontrar o melhor agrupamento para os nós sensores, definir o *cluster head* de cada agrupamento, construir a rede de comunicação e encontrar a rota de menor caminho (tempo) para o agente móvel. Dessa forma tem-se dois problemas bem conhecidos: o problema de roteamento e o problema de agrupamento dos nós sensores. Ambos são resolvidos de forma integrada, permitindo controlar o tempo de percurso da rota do agente móvel e, ao mesmo tempo, minimizar a energia total gasta na transmissão e recepção dos dados.

Este problema é modelado como uma variação do Problema da Árvore Geradora Mínima com Restrição de Saltos integrado com o Problema do Caixeiro Viajante. O modelo aqui proposto consiste em encontrar uma rede de comunicação entre os nós sensores e a rota do agente móvel de forma que todos os cluster heads sejam visitados, minimizando a energia total da rede. Para aprimorar a Qualidade de Serviço (QoS), o número de saltos é restrito, para economizar energia e diminuir a probabilidade de falhas na retransmissão dos dados. Por outro lado, para diminuir o atraso na entrega dos dados, limita-se o tempo de percurso da rota do agente móvel. Este problema, denominado Problema Integrado de Agrupamento e Roteamento com Restrição de Saltos e Tempo (PARST), é formulado por um modelo de Programação Linear Inteira Mista (PLIM).

Cada sensor coleta k bits na área de sensoriamento e envia seus dados diretamente para outro sensor: ou para um nó sensor comum da rede, ou para um cluster head, ou para o agente móvel, que é o caso dos cluster heads. Dessa forma, considera-se que a energia gasta pelo nó sensor é proporcional à quantidade de dados transmitidos e recebidos. Além disso, a energia gasta também está diretamente relacionada a distância de envio dos dados, quanto maior a distância maior o consumo de energia. Um modelo de energia que se

enquadra nos requisitos da RSSF deste trabalho é o modelo proposto por Heinzelman *et al.* (2002), por isso tal modelo foi adaptado e incorporado na formulação matemática do PARST.

Baseado no modelo matemático, um algoritmo exato é usado para resolver o problema. Entretanto, o tal algoritmo não se mostrou eficiente para resolver o problema, principalmente para instâncias com muitos nós sensores, uma vez que seu tempo de execução foi muito alto e em muitos casos não conseguiu sequer uma solução viável em 12 horas de execução. Com isso, uma abordagem híbrida para o PARST foi desenvolvida a fim de prover soluções viáveis (idealmente de boa qualidade) em tempos de execução aceitáveis, para permitir seu uso nos processos de decisão em RSSFs.

Esta abordagem híbrida é baseada em duas metaheurísticas: Algoritmo Genético e GRASP. Ambas são responsáveis por definir o melhor conjunto de cluster heads. Para avaliar cada solução gerada pelas metaheurísticas, heurísticas foram desenvolvidas com o objetivos de construir e avaliar a rede de comunicação e a rota do agente móvel considerando os cluster heads previamente definidos. Uma vez que tais heurísticas podem não conseguir ótimos locais, mecanismos de Busca Local são propostos.

Para verificar a qualidade das soluções encontradas pelo AG e GRASP, elas foram usadas como soluções iniciais no algoritmo exato. Ambas conseguiram encontrar soluções ótimas para algumas das instâncias com poucos nós sensores. Para instâncias maiores, a qualidade da solução não pode ser confirmada, mas uma solução viável pode ser encontrada em minutos, tanto pelo AG quanto pelo GRASP, enquanto o método exato não foi capaz de melhorar nem provar a otimalidade da solução inicial em 12 horas de execução.

Pode-se observar que usar um agente móvel na rede é uma estratégia vantajosa. O projetista da RSSF pode decidir o que é mais importante para sua rede: diminuir o gasto de energia, tendo um maior atraso na entrega dos dados, ou diminuir o atraso na entrega dos dados, mas aumentando o gasto energético da rede.

Finalmente, foi proposta uma abordagem baseada em geração de colunas para aumentar o número de rounds que a RSSF estará em funcionamento. O subproblema é resolvido em parte pelo método híbrido e quando este não for capaz de gerar uma coluna com custo reduzido positivo a formulação exata do subproblema é usada. Dessa forma, consegue-se ter um aumento considerável no número de rounds da rede em um tempo computacional aceitável. Considerando o Algoritmo Genético como método híbrido para o subproblema, chega-se a um ganho de mais de 5 vezes no número de rounds. Já o maior ganho do GRASP, como subproblema, foi de 4,8 vezes, mas o tempo de execução médio para o GRASP foi bem menor, se comparado com o tempo médio do AG.

6.2 Trabalhos futuros

O problema aqui estudado apresenta várias oportunidade em diferentes linhas de pesquisas. Para a formulação exata do PARST, podem ser explorados outros algoritmos exatos visando melhorar os limites inferiores e superiores obtidos, gastando um tempo computacional aceitável.

Outra alternativa seria uma modelagem multi-objetivo do problema, gerando um conjunto não dominante de soluções, entre energia consumida e o atraso na entrega dos dados. Com as soluções pareto, o projetista da rede poderia decidir o que é melhor para a aplicação da RSSF.

Para melhorar o tempo de vida da rede e diminuir ainda mais o atraso na entrega dos dados, pode-se usar mais de um agente móvel. Assim, o atraso e o consumo de energia poderiam ser reduzidos. Na formulação matemática do problema, bastaria acrescentar uma dimensão nas variáveis relacionadas à rota do agente móvel. Já no método híbrido, apenas a heurística construtiva para a rota do agente móvel seria modificada, agora construindo mais de uma rota.

Outra forma de aumentar a economia de energia seria considerar a cobertura da rede, uma vez que alguns nós sensores podem estar cobrindo a mesma área. Dessa forma, os nós sensores que cobrem uma área comum revezariam o sensoriameto, ficando ativo ou inativo por algumas rodadas.

6.3 Publicações

O trabalho desta dissertação gerou, até o momento, as seguintes publicações:

IEEE Congress on Evolutionary Computation (IEEE CEC 2013)

- *Título:* Lifetime maximization of hop-and-delay constrained wireless sensor networks with mobile agent
- *Autores:* Romão, O. C.; Santos, A. G.; Mateus, G. R.

Metaheuristics International Conference 2013 (MIC 2013)

- *Título:* Minimizing Energy Consumption on a Hop-contrained WSN using a Delay-constrained Mobile Agent
- *Autores:* Romão, O. C.; Santos, A. G.; Mateus, G. R.

Simpósio Brasileiro de Pesquisa Operacional (XLV SBPO)

- *Título:* An exact formulation and a hybrid heuristic method for Hop-contrained WSN using Delay-constrained Mobile Agent
- *Autores:* Romão, O. C.; Santos, A. G.

Referências Bibliográficas

Aioffi, W., Mateus, G. R., & Quintão, F. (2007). Optimization issues and algorithms for wireless sensor networks with mobile sink. In *International Network Optimization Conference*.

Aioffi, W. M. (2007). Métodos integrados para organização de rede de sensores sem fio com sorvedouro móvel e controle de densidade. Dissertação de mestrado, UFMG, Departamento de Ciência da Computação, Belo Horizonte - MG.

Akkaya, K. & Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3, 325–349.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38, 393–422.

Al-Karaki, J. N. & Kamal, A. E. (2004). Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6), 6–28.

Araujo, R. P., dos Santos, A. G., & Arroyo, J. (2009). Genetic algorithm and local search for just-in-time job-shop scheduling. In *IEEE Congress on Evolutionary Computation, 2009. CEC '09.* (pp. 955–961).

Arduino (2013). *Arduino no Piauí: Introdução a Redes de Sensores Sem Fio.* Disponível em: http://www.arduinopi.net/2010/10/introducao-redes-de-sensores-sem-fio.html Acesso em: 10 junho 2013.

Bechelane, C., Cunha, A. S., & Mateus, G. R. (2009). The minimum cost hop-and-root constrained forest in wireless sensor networks. *Electronic Notes in Discrete Mathematics*, 35(0), 139 – 144.

Bechelane, C. O. (2009). Uma Abordagem Para Minimização De Consumo De Energia Em Redes De Sensores Sem Fio Com Sorvedouros Móveis. Dissertação de mestrado, UFMG, Departamento de Ciência da Computação, Belo Horizonte - MG.

Behdani, B., Yun, Y., Smith, J. C., & Xia, Y. (2012). Decomposition algorithms for maximizing the lifetime of wireless sensor networks with mobile sinks. *Computers & Operations Research*, 39(5), 1054–1061.

Brittes, M. P. (2007). Uma proposta para melhoria de desempenho do protocolo LEACH para RSSF. Dissertação de mestrado, UTFPR, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Curitiba - PR.

Cai, W., Chen, M., Hara, T., Shu, L., & Kwon, T. (2011). A genetic algorithm approach to multi-agent itinerary planning in wireless sensor networks. *Mobile Networks and Applications*, 16(6), 782–793.

Cardei, M., Maccallum, D., Cheng, M., Min, M., Jia, X., Li, D., & Du, D. (2002). *Wireless Sensor Networks with Energy Efficient Organization*. Technical report, University of Minnesota, Department of Computer Science and Engineering, Minneapolis.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6), 791–812.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, 2, 393–410.

ECE (2013). *First of its kind Network Testbed*. Electrical and Computer Engineering - Virginia Tech.
Disponível em: http://www.ece.vt.edu/news/fall05/sensornetwork.htm Acessado em: 20 maio 2013.

Feo, T. A. & Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2), 67–71.

Gouveia, L. (1996). Multicommodity flow models for spanning trees with hop constraints. *European Journal of Operational Research*, 95(1), 178 – 190.

Heidari, E. & Movaghar, A. (2011). An efficient method based on genetic algorithms to solve sensor network optimization problem. *CoRR*, abs/1104.0355.

Heinzelman, W. B., Chandrakasan, A. P., & Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4), 660 – 670.

Heinzelman, W. R., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000.*, HICSS '00 (pp. 8020–8029). Washington, DC, USA: IEEE.

Heinzelman, W. R., Kulik, J., & Balakrishnan, H. (1999). Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99 (pp. 174–185). New York, NY, USA: ACM.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press.

Huang, C. & Tseng, Y. (2003). The coverage problem in a wireless sensor network. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, WSNA '03 (pp. 115–121). New York, NY, USA: ACM.

ILOG CPLEX Optimization, I. (2011). *IBM ILOG CPLEX Optimization Studio - CPLEX User's Manual*.

Jong, K. A. & Spears, W. M. (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1), 1–26.

Keskin, M. E., Altinel, I. K., Aras, N., & Ersoy, C. (2011). Lifetime Maximization in Wireless Sensor Networks Using a Mobile Sink with Nonzero Traveling Time. *The Computer Journal*, 54(12), 1987–1999.

Kim, H. S., Abdelzaher, T. F., & Kwon, W. H. (2003). Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03 (pp. 193–204). New York, NY, USA: ACM.

Lindsey, S. & Raghavendra, C. (2002). Pegasis: Power-efficient gathering in sensor information systems. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 3 (pp. 3–1125–3–1130 vol.3).

Meguerdichian, S. & Potkonjak, M. (2003). *Low Power 0/1 Coverage and Scheduling Techniques in Sensor Networks*. Technical report, Computer Science Department, University of California Los Angeles.

Nakamura, F. G., Quintão, F. P., Menezes, G. C., & Mateus, G. R. (2005). An Optimal Node Scheduling for Flat Wireless Sensor Networks. *4th International Conference on Networking*, 3420, 475–482.

Park, M. W., Choi, J. Y., Han, Y. J., & Chung, T. M. (2009). An energy efficient concentric clustering scheme in wireless sensor networks. In *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on* (pp. 58–61).

Park, S., Savvides, A., & Srivastava, M. B. (2001). Simulating networks of wireless sensors. In *Proceedings of the Winter Simulation Conference, 2001*, volume 2 (pp. 1330–1338 vol.2).

Poli, R. & Langdon, W. B. (1998). Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation*, 6, 231–252.

Rappaport, T. (2001). *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2nd edition.

Santos, A. G. (2008). *Método de Geração de Colunas e Meta-heurísticas para Alocação de Tripulação*. Tese de doutorado, UFMG, Departamento de Ciência da Computação, Belo Horizonte - MG.

Seo, H., Oh, S., & Lee, C. (2009). Evolutionary genetic algorithm for efficient clustering of wireless sensor networks. In *Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference*, CCNC'09 (pp. 258–262). Piscataway, NJ, USA: IEEE Press.

Siqueira, I. G., Figueiredo, C. M. S., Loureiro, A. A. F., Nogueira, J. M., & Ruiz, L. B. (2006). An integrated approach for density control and routing in wireless sensor networks. In *20th International Parallel and Distributed Processing Symposium* (pp. 10–19).

Slijepcevic, S. & Potkonjak, M. (2001). Power Efficient Organization of Wireless Sensor Networks. *IEEE International Conference on Communications*, 2, 472–476.

Song, L. & Hatzinakos, D. (2007). Architecture of wireless sensor networks with mobile sinks: Sparsely deployed sensors. *IEEE Transactions on Vehicular Technology*, 56(4), 1826–1836.

Valle, C. A. (2009). Algoritmos de otimização para roteamento e agrupamento em redes de sensores sem fio com sorvedouros móveis. Dissertação de mestrado, UFMG, Departamento de Ciência da Computação, Belo Horizonte - MG.

Wang, W., Srinivasan, V., & Chua, K. C. (2005). Using mobile relays to prolong the lifetime of wireless sensor networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking* (pp. 270–283). New York, NY, USA: ACM.

Wifly (2013). *Wifly: Virtual coordinates, mobile sink and R/C planes*.
Disponível em: http://www.eecs.berkeley.edu/˜ watteyne/wifly.html Acesso em: 25 maio 2013.

Wu, Q., Rao, N. S. V., Barhen, J., Iyengar, S. S., Vaishnavi, V. K., Qi, H., & Chakrabarty, K. (2004). On computing mobile agent routes for data fusion in distributed sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, 16(6), 740–753.

XBOW (2013). *MICA2*.
Disponível em: http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=174 Acesso em: 20 maio 2013.

Yun, Y. & Xia, Y. (2010). Maximizing the Lifetime of Wireless Sensor Networks with Mobile Sink in Delay-Tolerant Applications. *IEEE Transactions on Mobile Computing*, 9, 1308–1318.

Zheng, J. & Jamalipour, A. (2009). *Wireless Sensor Networks: A Networking Perspective*. Hudson, New Jersey, USA: Wiley-IEEE Press.